Multilevel techniques for Reservoir Simulation

Christensen, Max la Cour

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

Multilevel techniques for Reservoir Simulation

Max la Cour Christensen

Kongens Lyngby 2016

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary (English)

The subject of this thesis is the development, application and study of novel multilevel methods for the acceleration and improvement of reservoir simulation techniques. The motivation for addressing this topic is a need for more accurate predictions of porous media flow and the ability to carry out these computations in a timely manner. This will lead to better decision making in the production of oil and gas. The goal is attained in various ways throughout the thesis work. Specifically, three fields of multilevel methods have been addressed in this work, namely

- Nonlinear multigrid (the Full Approximation Scheme)
- Variational (Galerkin) upscaling
- Linear solvers and preconditioners

First, a nonlinear multigrid scheme in the form of the Full Approximation Scheme (FAS) is implemented and studied for a 3D three-phase compressible rock/fluids immiscible reservoir simulator with a coupled well model. In a fair way, it is compared to the state-of-the-art solution scheme used in industry and research simulators. It is found that FAS improves time-to-solution by having a larger basin of attraction, faster initial convergence, data locality and a lower memory footprint. The study is extended to include a hybrid strategy, where FAS is combined with Newton's method to construct a multilevel nonlinear preconditioner. This method demonstrates high efficiency and robustness.

Second, an improved IMPES formulated reservoir simulator is implemented using a novel variational upscaling approach based on element-based Algebraic

Multigrid (AMGe). In particular, an advanced AMGe technique with guaranteed approximation properties is used to construct a coarse multilevel hierarchy of Raviart-Thomas and L^2 spaces for the Galerkin coarsening of a mixed formulation of the reservoir simulation equations. By experimentation it is found that the AMGe based upscaling technique provided very accurate results while reducing the computational time proportionally to the reduction in degrees of freedom. Furthermore, it is demonstrated that the AMGe coarse spaces (interpolation operators) can be used for both variational upscaling and the construction of linear solvers. In particular, it is found to be beneficial (or even necessary) to apply an AMGe based multigrid solver to solve the upscaled problems. It is found that the AMGe upscaling changes the spectral properties of the matrix, which renders well-known state-of-the-art solvers for this type of system useless.

Third, FAS is combined with AMGe with guaranteed approximation properties to obtain a nonlinear multigrid solver for unstructured meshes. The FAS-AMGe solver is applied to a simplistic but numerically challenging mixed (velocity/-pressure) model for porous media flow. In a fair way, FAS-AMGe is compared to Newton's method and Picard iterations. It is found that FAS-AMGe is faster for the cases considered.

Finally, a number of multigrid linear solvers and preconditioners are implemented for various linear systems. In particular AMGe are used in the construction of multigrid preconditioners. These are compared to two state-of-the-art block diagonal preconditioners based on 1) a Schur complement with an Algebraic Multigrid (AMG) solver and 2) an augmented Lagrangian formulation using the Auxiliary Space AMG solver.

In addition to the research mentioned above, a sequential in-house COmpositional reservoir SImulator (COSI) with many features is parallelized in a distributed setting (MPI) using the PETSc framework. A parallel preconditioner based on the Constrained Pressure Residual method, Algebraic Multigrid and Restricted Additive Overlapping Schwarz with Incomplete LU solves on each subdomain is implemented. It is found that switching the traditionally used method, namely parallel ILU, with Restricted Additive Overlapping Schwarz results in a significant increase in parallel scalability while still maintaining similar robustness and efficiency.

Summary (Danish)

Emnet for denne afhandling er udvikling og anvendelsen af nye multilevel metoder til accelerering og forbedring af reservoir simulerings teknikker. Motivationen for at adressere dette emne er et behov for mere nøjagtige forudsigelser af flow i porøse medier og for at være i stand til at udføre de nødvendige beregninger indenfor en rimelig tid. Dette vil føre til bedre beslutningstagen i produktionen af olie og gas. Målet opnås på forskellige måder i arbejdet udført i forbindelse med denne afhandling. Helt specifikt er der blevet arbejdet indenfor 3 forskningsområder for multilevel metoder, nemlig

- Ikke-lineære multigrid teknikker (Full Approximation Scheme)
- Variational (Galerkin) opskalering
- Linæere løsere og prækonditionering

Som et første skridt er en ikke-lineær multigrid løser i form af Full Approximation Scheme (FAS) implementeret og studeret for en 3D, tre-fase, kompressibel sten/væske, ikke-blandbare (væsker) reservoir simulator med en koblet brøndmodel. På et fair grundlag er denne løsningstrategi blevet sammenlignet med den allerede eksisterende nyeste og bedste løsningstrategi, som bliver brugt i industrielle og forsknings simulatorer. Resultaterne viser, at FAS forbedrer beregningstiden ved at have et større basin of attraction, hurtigere initial konvergens, data lokalitet og et mindre hukommelsesbehov. Studiet er udvidet til også at indebære en hybrid strategi, hvor FAS er kombineret med Newton's metode, så man får en multilevel ikke-lineær prækonditioner. Denne metode viser sig at være meget effektiv og robust.

I andet skridt er en forbedret IMPES formuleret reservoir simulator implementeret baseret på en ny variational opskaleringsmetode, som bygger på elementbaseret Algebraisk Multigrid (AMGe). Helt specifikt er der blevet brugt en avanceret AMGe teknik med garanterede approksimeringsegenskaber til at konstruere et groft multilevel hierarki af Raviart-Thomas og L^2 rum til Galerkin opskalering af en mixed formulering of reservoir simulerings ligningerne. Gennem eksperimenter er det blevet fastslået, at den AMGe baserede opskaleringsteknik giver meget nøjagtige resultater samtidig med at den reducerer beregningstiden propertionelt med reduktionen i frihedsgrader. Yderligere bliver det demonstreret, at de AMGe grove rum (interpolerings operatorer) kan bruges både til variational opskalering og til konstruktionen af lineære løsere. Specifikt er det blevet påvist at være fordrende (eller faktisk nødvendigt) at anvende en AMGe baseret multigrid løser til at løse de opskalerede problemer. Dette skyldes, at AMGe opskaleringen ændrer de spektrale egenskaber af matricen, hvilket gør, at velkendte state-of-the-art løsere til den type problemer er ubrugelige.

Som et tredje skridt er FAS blevet kombineret med AMGe med garanterede approksimeringsegenskaber for at få en ikke-lineær multigrid løser til ustrukturerede net. FAS-AMGe løseren er anvendt på et simpelt men numerisk udfordrende mixed (hastighed/tryk) model for flow i porøst medium. På et fair grundlag er FAS-AMGe blevet sammenlignet med Newton's metode og Picard iterationer. Det viser sig, at FAS-AMGe er hurtigere for de undersøgte problemer.

Slutteligt er der blevet implementeret en række multigrid lineære løsere og prækonditionere til forskellige lineære ligningssystemer. For eksempel er AMGe blevet brugt til at konstruere multigrid prækonditionere. Disse er blevet sammenlignet med to state-of-the-art blok diagonal prækonditionere baseret på 1) et Schur komplement med en Algebraisk Multigrid (AMG) løser og 2) en augmenteret Lagrangian formulering, som bruger Auxiliary Space AMG løseren.

Udover den forskning der er nævnt opover, er en sekventiel in-house kompositionel reservoir simulator (COSI) med mange features blevet paralleliseret i en distribueret setting (MPI) ved brug af PETSc. En parallel prækonditioner baseret på Contrained Pressure Residual metoden, Algebraisk Multigrid og Restricted Additive Overlapping Schwarz med Incomplete LU løsere på hvert subdomæne er blevet implementeret. Det viser sig, at en udskiftning af den ellers normalt anvendte metode, nemlig parallel ILU, til fordel for Restricted Additive Overlapping Schwarz resulterede i en signifikant øgning af parallel skalering samtidig med at bibeholde et tilsvarende niveau af robusthed og effektivitet.

Preface

This thesis was prepared at Department of Applied Mathematics and Computer Science (DTU Compute), Technical University of Denmark in partial fulfillment of the requirements for receiving the Ph.D degree. The work presented in this thesis was carried out from April 2013 to June 2016. The work has been financed through the Industrial PhD programme at Innovation Fund Denmark with the company Lloyd's Register Consulting as primary sponsor. Specifically, I have been employed at Lloyd's Register Consulting and enrolled at the Technical University of Denmark.

Parts of the work has been carried out during several research visits at the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory. In total, around 1 year was spent at Lawrence Livermore National Laboratory. Furthermore, I had a shorter research visit to The University of Texas at Austin. During and in between research visits, in addition to the supervision by Associate Professor Allan P. Engsig-Karup, I have received supervision and mentoring by Professor Panayot S. Vassilevski and Dr. Umberto Villa. Finally, Dr. Stefan Lemvig Glimberg from Lloyd's Register Consulting has been the industry supervisor. The thesis is structured as an article-based thesis, where four of the chapters are articles either submitted, published or presented at a conference.

Kongens Lyngby, June 2016

Max la Cour Christensen

Acknowledgments

A number of people have provided their help and support, for which I am very grateful. First of all I would like to thank my advisor at DTU, Allan P. Engsig-Karup for his guidance and many valuable inputs but most of all for allowing me the freedom to pursue my own ideas. Furthermore, I would like to thank my company advisor Stefan Lemvig Glimberg for our collaboration in parallelizing COSI and for making the many hours of coding (and debugging) more fun. Finally I would also like to thank my friends and colleagues in the Fluid Dynamics team at Lloyd's Register Consulting for providing a great and fun work environment.

During my Ph.D. I was fortunate to have the opportunity to spend around 1 year at Lawrence Livermore National Laboratory. A very special thanks to Panayot Vassilevski for making this possible. I am very grateful for all that I have learned by working with him and his group at LLNL, and as the thesis reveals, this collaboration has had a huge impact on the direction of my research. In particular at LLNL, I would like to thank Umberto Villa for his friendship and mentoring both during my stays at LLNL and when I was in Denmark.

Finally, I wish to thank my family, friends and my partner in life Sofie for their patience during this period and their continuous support.

Contents

Chapter 1

Introduction

The purpose of this introductory chapter is to motivate the work presented in this thesis, introduce basic multigrid ideas to set the scene for the more advanced methods presented later and finally to give an overview of the key contributions.

1.1 Motivation and objectives

The purpose of the work presented in this thesis is to enable better predictive capabilities for porous media flow applications. Simulation of the flow of oil, gas and water in the subsurface remains a challenging task with many uncertainties, which may affect the reliability of the resulting production forecasts. This may be caused by inaccurate or lacking data, unsatisfactory resolution of the simulations, inaccurate numerical techniques, poor description of the physical processes, etc. The failure to accurately predict oil and gas production may result in non-optimal decisions with a risk of very significant financial losses. From a more general point of view, some of the techniques investigated in this thesis may find relevance in other fields of engineering, where similar issues can be found.

The following classes of methods have been investigated to improve upon the issues mentioned above.

- *Optimal linear solvers/preconditioners and nonlinear solvers.* Increasing the resolution of simulations is one way to provide more accurate predictions. In computational fluid dynamic studies, depending on the complexity of the model, it is common practice to ensure a mesh-independent solution. Unfortunately this is not the case in reservoir simulation studies, where smaller simulation models often are employed to enable a reasonable time-to-solution. Increasing the size of the simulation models is only viable if the underlying solvers are algorithmically scalable. That is, the computational time of a simulation should be linearly proportional to the number of grid cells/elements. To the knowledge of the author, the only way to obtain this is via the application of multigrid techniques. Many advances in multigrid techniques for porous media flow have been made, however there are still challenges with obtaining a truly scalable solver scheme for reservoir simulation capable of handling the difficulties related to highly heterogeneous permeability/porosity fields, strong nonlinearities and point sources introduced by wells. This is largely attributed to the fact that the system of partial differential equations contains transport equations, which exhibit local saturation fronts difficult to sufficiently capture in a multigrid scheme.

- *Accurate numerical upscaling techniques.* In many scenarios it can be beneficial to base costly and important decisions on a sound statistical basis rather than making these decisions based on a single deterministic calculation. Methods such as Monte Carlo simulations can be used to provide this sound statistical basis, however these methods require the simulation of many permutations of the model. In the context of 3D discretization of time-dependent nonlinear systems of partial differential equations, it quickly becomes impossible to carry out the required number of simulations. To remedy this issue, coarse representations of the underlying model can be used to accelerate the Monte Carlo simulations. This technique has been denoted as Multilevel Monte Carlo. The work presented in this thesis deals with methods capable of providing highly accurate coarse representations of the model. This is achieved through the application of a certain class of numerical upscaling techniques, where instead of coarsening (upscaling) the coefficients of the partial differential equations, it is the system of partial differential equations itself, which is upscaled. This technique allows for more accurate upscaling, which in turn results in an improved acceleration of the Multilevel Monte Carlo method. In addition to being applicable for uncertainty quantification, this numerical upscaling technique also holds great potential in the acceleration of optimization techniques, where again many realizations of the model is needed. For instance, it can be used to accelerate the optimization of well placements, water and gas sweeping patterns, etc. Furthermore as wells become more advanced and allow more control of inflow/outflow in individual segments

of the wells, we see an increasing interest in real time production optimization. To accommodate this would require very fast simulation techniques.

- *Alternative formulations and discretization methods.* Alternative formulations and discretization methods have been sought out in an attempt to find a scheme more applicable for the previously mentioned numerical upscaling techniques and where the resulting systems can be solved more easily by multigrid techniques. In particular, so-called mixed formulations are investigated, where in addition to solving for pressure and saturations, the velocity field is also solved for.
- *Methods suitable for large-scale parallelization on clusters.* With a stagnation in processor clock frequency, the only way currently to achieve speedups is through the use of parallel computers. Therefore, as a commonality for all techniques studied in this work, a lot of emphasis is put on finding methods suitable for parallelization on clusters. This is largely an exercise in introducing methods with potential for parallelization and extracting the fine grained parallelism from the algorithms.

1.2 Literature review

Since the thesis work spans several fields such as

- Nonlinear solvers
- Variational upscaling
- Linear solvers/preconditioners

the amount of prior research is enormous. In this section, only the most relevant papers for the particular methods investigated in this thesis are reviewed. Naturally, this section will contain overlaps with the literature studies in each of the four papers included in this thesis.

1.2.1 Nonlinear solvers

Flow of oil, water and gas in porous media can be described mathematically with a system of partial differential equations (PDEs), [16, 30]. With conventional techniques, it is common to use a global linearization Newton-type method

to solve the strongly nonlinear system of equations arising from the spatial and temporal discretization of the governing equations [16]. This global linearization results in large linear systems, and hence the linear solver component typically constitutes the majority of the computational time of a simulation. Iterative linear solvers depend on efficient preconditioners, which can be difficult to parallelize to the extent required by many-core hardware [104]. Additionally, the memory required to store the sparse Jacobian for the linear systems is significant. This is not in line with modern hardware trends, which indicate that memory continues to be limited per core.

The Full Approximation Scheme (FAS) is a multigrid method for nonlinear problems, [24, 50, 109, 51]. Its most widespread use is in geometric multigrid on structured grids due to difficulties associated with defining a coarse nonlinear operator on unstructured meshes. On unstructured grids, the most popular choice of nonlinear solver schemes is typically Newton-Krylov methods preconditioned by e.g. a black box method such as Algebraic Multigrid (AMG), [23, 102]. However, FAS offers potential benefits with respect to traditional methods, such as a larger basin of attraction, faster initial convergence, data locality and lower memory footprint. Several papers have addressed the application of FAS to unstructured grids. In [83, 84], FAS based on agglomeration multigrid is compared to Newton-Multigrid. In these papers, coarse grid control-volumes are formed by merging together finer grid control-volumes. Based on this agglomeration of control-volumes, the associated interpolators between grids are defined as simple injection/piecewise constants. In a multilevel context, piecewise constant interpolation between grids is insufficient and will result in loss of accuracy and therefore loss of performance in the overall multigrid scheme, [82]. An improvement was suggested in [82] to use an implicit prolongation operator, however, it may be too expensive to be worth the gain in convergence rate.

Whilst FAS have been implemented and modified for the Navier-Stokes equations [56], interestingly, very little work has been published on the use of the FAS method for reservoir simulation. In [86] convergence rates of two variations of FAS on a simple 2D immiscible two-phase homogeneous structured grid example without gravity are demonstrated. Specifically, it is demonstrated that FAS provides fast, grid-independent convergence behaviour and optimal complexity, implying the computational cost per time step per grid point is independent of the number of grid points.

FAS has previously been combined with AMGe to obtain a nonlinear solver for lowest order nodal finite elements, [38, 62]. Mesh-independent convergence was demonstrated for an elliptic 2D model problem, [38]. In [38], the method is based on the AMGe introduced in [63, 113]. This results in coarse spaces, where only one degree of freedom can be used for each agglomerate. Consequently, it is difficult to maintain accuracy on very coarse agglomerate meshes, resulting

in a degradation of the FAS solver performance.

1.2.2 Variational upscaling

Upscaling of geological properties is an essential practice in reservoir simulation, since the spatial resolution of the geological model often is too high for reservoir simulators to execute in practical times. The traditional approach employed today resorts to computing effective properties of the subsurface (permeability/porosity) by homogenization (averaging) techniques. Homogenization is formally an averaging of processes (and or mathematical operators), and hence, it goes far beyond averaging of parameters. Many techniques are available for homogenization. The so-called flow-based upscaling methods are among the most used ones. They are typically based on solving a simple steady-state elliptic differential equation. Given the solution of this equation, effective coarse permeabilities can be computed. Some flow-based upscaling methods provide full tensor coarse permeabilities, but in practice mostly diagonal tensor permeabilities are used. These effective coarse properties are then perceived as the "true" model from this point on. However, the use of homogenization in the workflow introduces a black box step, where the relation/difference between the solution of the upscaled model and the solution of the fine grid model (geological resolution) is difficult (or impossible) to determine. A significant body of research has gone into improving this workflow by introducing new upscaling methods, which take into account more information of the underlying problem. In the petroleum engineering community, these methods are typically referred to as multiscale methods. In the following, references and comments are given for the papers closest to the approach to variational upscaling used in this thesis.

The methods described in this thesis are strongly related to the Multiscale Mixed Finite Element Method (MsMFEM). MsMFEM stems from early work described in [52] and [31], where specific finite-element basis functions were used to construct a tool for multiscale solution of elliptic partial differential equations in both primal and mixed form. Since then, much research has been carried out on this topic to improve the approximation properties and extend the range of physical phenomena described by the models, [10, 12, 11, 5, 70]. Among other things, the method was extended to achieve locally mass conservative velocity fields on the subgrid scale, which enabled a combination of MsMFEM and streamline simulations, [7]. Other work focuses on updating the multiscale basis functions for time dependent problems [69] or to capture specific features of the flow [6]. Multiscale methods have also attracted attention for locally conservative mimetic finite difference methods, [8], and for finite volume methods, [59, 39]. Adaptive strategies for multiscale techniques have also been proposed,[39, 75].

Multiscale methods have been extended for mimetic finite differences to work in a multilevel way for two-phase flow problems, [74, 75, 76]. Multiscale multilevel mimetic methods, namely M^3, have several similarities to the AMGe approach described in this work, and a few differences. Important similarities include the ability to handle aggregates with non-planar faces, and achieve local mass conservation on all levels, and provide support for well models. The AMGe approach possesses all these properties with the additional flexibility to assign a variable number of degrees of freedom per agglomerated face (interface between two agglomerated elements) that is automatically determined by the desired accuracy and by the topology of the agglomerated face by means of SVD, [72]. Finally, the multilevel upscaling technique introduced in [80] leverages the components from a multigrid algorithm, using algebraically constructed coarse spaces and variational Galerkin coarsening. The AMGe approach used in this work exhibits similar features with the additional caveat that the coarse spaces maintain guaranteed order of approximation at all levels, [72].

In the algebraic multigrid (AMG) community, the construction of coarse problems was an essential component to develop efficient multilevel solvers for the fine-grid (fine-scale) problems of interest, especially in the unstructured mesh setting. It was recognized for quite some time, that an efficient two-grid (TG) solver requires as a necessary condition a coarse space that admits certain weak approximation properties. For a rigorous proof of this fact, see [43]. This fact can be viewed as a cornerstone motivation for using AMG-constructed finite element coarse spaces as discretization spaces, i.e., as a tool for numerical upscaling. The interested reader is referred to the overview in [115] for more details. For some early work on using operator-dependent (AMG) coarse spaces for numerical homogenization, see [68].

The focus of research in AMGe turned to constructing coarse spaces, which can handle general classes of finite element spaces, and hence be applicable to broad classes of PDEs. This resulted in methods for constructing coarse spaces that form a de Rham complex (i.e. the sequence of H^1-conforming, $H(\text{curl})$-conforming, $H(\text{div})$-conforming and L^2-conforming spaces) with applications to elliptic PDEs, Maxwell equations, Darcy flow equations, etc, [99]. The work [99], although specifically motivated to construct coarse de Rham complexes for use in multigrid solvers, provided the basis for extensions finalized in [72], to build coarse spaces with guaranteed approximation properties, giving rise to an efficient upscaling tool. Since the construction of coarse spaces applies to the entire de Rham sequence, the developed technique can also be used for other applications such as the mixed formulation of the Brinkman problem, [111, 116].

Often multiscale methods for reservoir simulation solve for the pressure (and velocity) on a coarse scale and keep the saturation equations on the fine grid. With this approach, solving the saturation equations quickly becomes the dom-

inant bottleneck. Methods have been developed to also upscale the saturation equation, [120, 41]. Utilizing AMGe enables upscaling of not only the mixed system for velocity and pressure, but also the transport equations for the saturations using the same framework. For problems involving quantities of interest that do not require a fine-scale solution of the saturation equations, such fully upscaled models can greatly accelerate standard methods in uncertainty quantification (e.g. using Multilevel Monte Carlo [37, 67]) and optimization (e.g. MG/OPT [21]) due to the ability to simulate with good accuracy at different levels of resolution with a reduction in computational cost equivalent to the reduction in the degrees of freedom for the upscaled models. As final remarks, AMGe has been developed since its introduction ([61], [112], [27, 28], [71]) as a general multilevel coarsening framework with a wide range of applications. In addition to numerical upscaling, it is also designed to create efficient and optimal solvers that can be adapted throughout the simulation, [64]. Furthermore, the AMGe upscaling technique targets general unstructured meshes as well as higher-order elements. As such it is distinctly different (as being more general) than the above referred multiscale and mimetic methods.

1.2.3 Linear solvers/preconditioners

Conventional global linearization techniques (Newton's method and Picard iterations) results in large sparse linear systems, where the majority of the computational time is spent. The equations governing porous media flow can be formulated in various ways. In this work, two different types of formulations have been the topic of research. The first one, here called the standard formulation, is the one used in most commercial and research reservoir simulators. It results in a nonsymmetric M-matrix with mixed characteristics. The second formulation of focus in this work is the mixed formulation, which results in an indefinite saddle point problem, which depending on the formulation may be symmetric or nonsymmetric. These two types of matrices are distinctly different and require different solution strategies.

The linear solver methods originally employed for the standard formulation include preconditioned Krylov subspace methods such as ORTHOMIN, with nested factorization as the preconditioner [9]. A significant advance in preconditioning for the type of linear systems generated in a reservoir simulator was made with the Constrained Pressure Residual preconditioning (CPR) method [117, 55]. CPR preconditioning was developed specifically for reservoir simulation and acknowledges that the governing equations are of a mixed elliptic-hyperbolic type (in fact the equations may be parabolic, but still they each exhibit elliptic or hyperbolic type behaviour). By targeting the elliptic part of the system as a separate inner stage, the CPR method can achieve an improved

convergence rate for the complete linear systems. Whilst use of the CPR method is largely restricted to reservoir simulation it is worth noting that conceptually the method is similar to the SIMPLE-type schemes designed for the Navier-Stokes equations in which the pressure and velocity components of the solution are targeted separately [100]. The mixed formulation results in indefinite saddle point problems. A number of preconditioners and linear solvers have been developed for these. In particular, hybridization is a common solution strategy or block diagonal preconditioners based on Schur complements and algebraic multigrid. Specifically, we would like to highlight the Auxiliary Space AMG solver (ADS), [110]. For an overview of solvers for saddle point problems see [20].

1.3 Basic concepts of multigrid methods

This section introduces key concepts needed for the understanding of the multigrid methods described in the subsequent papers and chapters. Multigrid is a broad term applicable to many methods who share the same fundamental attributes. These attributes will be outlined in depth below. The most common use of multigrid methods is for the solution of linear systems such as

$$A\mathbf{u} = \mathbf{f}, \tag{1.1}$$

where A is a sparse matrix, $\mathbf{u}$ is the sought after solution vector and $\mathbf{f}$ is the known right hand side. Multigrid can either be applied directly to the linear system or as a preconditioner for other linear solvers (CG, GMRES, BiCGSTAB, etc). As it is evident from its name, multigrid uses a hierarchy of nested grid levels. This enables the construction of solvers capable of obtaining O(N) scaling. This hierarchy of grid levels is constructed such that $h < h_1 < h_2 < \dots h_L$, where h denotes the grid size of the finest grid and h_L denotes the grid size of the coarsest grid.

Multigrid relies on two operations to complement each other, namely

- relaxation and
- coarse-grid correction

The job of relaxation (also called smoothing) is to remove high-frequency errors. Removing high-frequency errors allows for an accurate interpolation of some set of values between grid levels. Figure 1.1 illustrates the result of applying Gauss-Seidel smoothing to a 2D Poisson problem, where the initial guess contains

random noise. Gauss-Seidel smoothing can be written in matrix form as $\mathbf{u}^{k+1} = -(D+L)^{-1}U\mathbf{u}^k + (D+L)^{-1}\mathbf{f}$, where D is the diagonal of A, L is the lower triangular portion of A and U is the upper triangular portion of A.

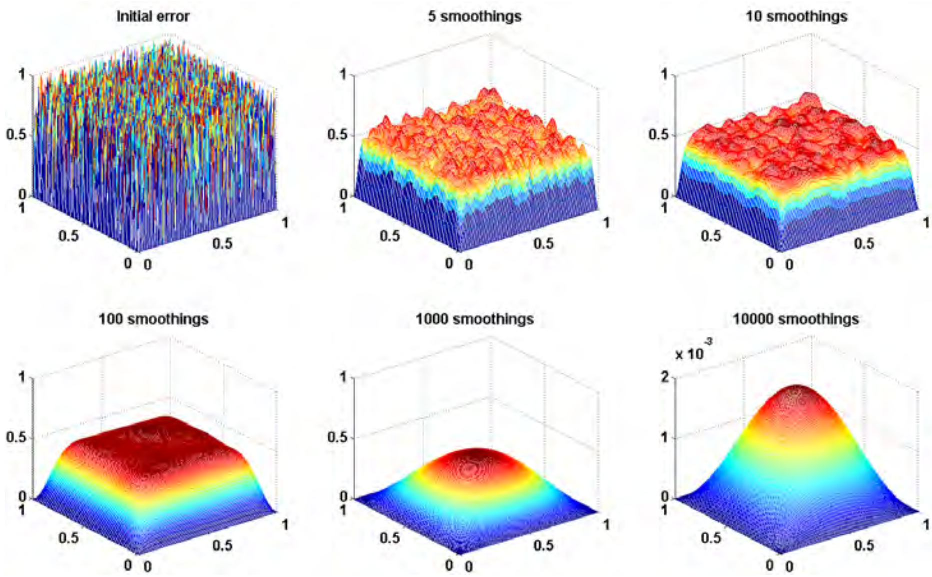


Figure 1.1: Gauss-Seidel smoothing on a Poisson problem with an initial guess containing random noise.

It is clear that with only few iterations of the Gauss-Seidel smoother, the high-frequency errors have been smoothed out significantly, however even after 100-1000 smoothings, the magnitude of the error is still roughly the same. More specifically, the low-frequency error still remains. To remedy this issue, multi-grid uses a coarse-grid correction. The principle of coarse-grid correction is to approximate the error on the coarse grids (where it is cheap to do smoothing), interpolate that error to a finer grid and correct the approximate solution. By carrying out computations on coarser grids, information is transported quickly through the domain. Coarse-grid correction should be designed such that it effectively removes the low-frequency errors, which have not been removed by the relaxation alone. The equation approximately solved on the coarse grids is the residual equation $A\mathbf{e} = \mathbf{r}$ which comes from

$$\begin{aligned} A\mathbf{u} = A(\mathbf{v}+\mathbf{e}) = A\mathbf{v} + A\mathbf{e} = f \quad &\Leftrightarrow \\ A\mathbf{v} + A\mathbf{e} = A\mathbf{v} + \mathbf{r} \quad &\Leftrightarrow \\ A\mathbf{e} = \mathbf{r} \quad & \end{aligned} \tag{1.2}$$

where $\mathbf{r} = \mathbf{f} - A\mathbf{v}$. Here $\mathbf{u} = \mathbf{v} + \mathbf{e}$ where $\mathbf{v}$ is an approximation to the solution $\mathbf{u}$ and $\mathbf{e}$ is the error. The error $\mathbf{e}$ is then interpolated to a finer grid, where it is used to correct $\mathbf{v}$. The multigrid algorithm is outlined in Algorithm 1.

Algorithm 1 - $\mathbf{v} = \text{MG\_Vcycle}(l, \mathbf{v}, \mathbf{f})$

1: **if** l == nLevels-1 (coarsest grid) **then**
2: Solve $\mathcal{A}_l \mathbf{v}_l = \mathbf{f}_l$ approximately or exact
3: **else**
4: Smoothing on $\mathcal{A}_l \mathbf{v}_l = \mathbf{f}_l$
5: Compute defect: $\mathbf{d}_l = \mathbf{f}_l - \mathcal{A}_l \mathbf{v}_l$
6: Restrict defect: $\mathbf{d}_{l+1} = I_l^{l+1} \mathbf{d}_l$
7: $\mathbf{e}_{l+1} = \text{MG\_Vcycle}(l+1, \mathbf{0}_{l+1}, \mathbf{d}_{l+1})$
8: Prolongate correction: $\mathbf{e}_l = I_{l+1}^{l} \mathbf{e}_{l+1}$
9: Correct approximation: $\mathbf{v}_l = \mathbf{v}_l + \mathbf{e}_l$
10: Smoothing on $A_l \mathbf{v}_l = \mathbf{f}_l$
11: **end if**

Applying the multigrid algorithm to the same Poisson problem as in Figure 1.1, the error between the analytical solution and the approximate solution is plotted in Figure 1.2. Only two sweeps of the Gauss-Seidel smoother was applied at each level. Notice how introducing the coarse-grid correction, makes the method able to reduce the error with one order of magnitude per multigrid iteration (V-cycle).

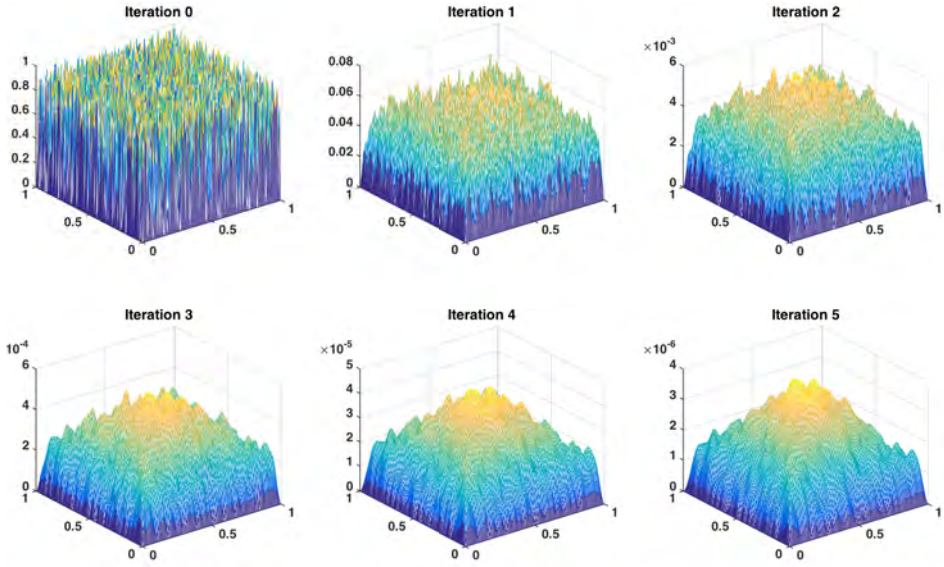


Figure 1.2: Error between analytical solution and approximate solution for multigrid with 2 Gauss-Seidel smoothing iterations on a Poisson problem with an initial guess containing random noise.

1.4 Geometric vs. algebraic multigrid

There are many different version of multigrid. The two broad classes of multigrid are geometric multigrid and algebraic multigrid. In geometric multigrid, the knowledge of the underlying grid is used for the construction of the multigrid components. Specifically, the smoother and interpolation operators are based on the grid structure. Typically these are based on stencil type operations. For instance, the interpolation from a fine to a coarse grid may be simple averaging of a number of cells into one coarse grid cell.

In algebraic multigrid, the grid is ignored and only the coefficients of the system matrix are used for the construction of the multigrid components. For instance, the interpolation operator is constructed by looking at the magnitude of the values in the system matrix and letting this dictate the important couplings to be preserved in the coarse system matrix. The coarse system matrix is typically found in a Galerkin sense, e.g. $A_H = P^T A_h P$. Here the interpolation

operator is given by the matrix P, which interpolates from a coarse level to a fine level. Figure 1.3 conceptually illustrates the differences between geometric and algebraic multigrid.

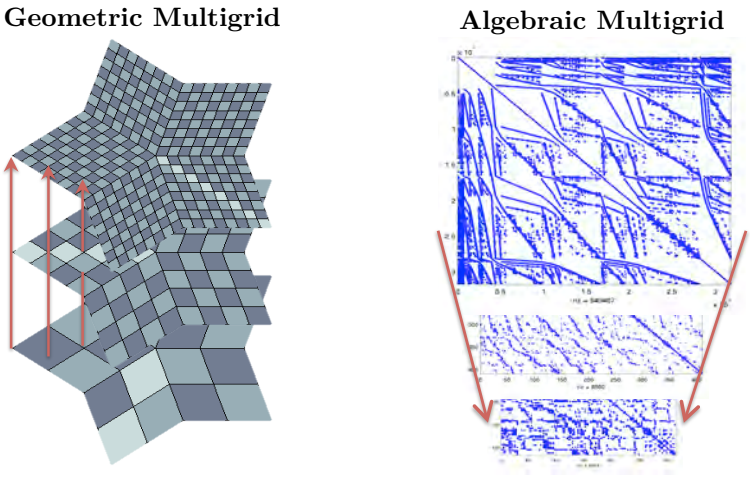


Figure 1.3: Illustration of differences between geometric and algebraic multigrid. Note that in these pictures, the coarse matrices on the right are in fact square. Also note that the arrows are unrelated to multigrid cycles, but rather meant as an illustration of some key points. In geometric multigrid, the fact that the arrows point upwards indicate that for general unstructured meshes, it may be impossible to generate a sequence of grids from fine to coarse. Instead the sequence of grids can be generated from coarse to fine by refinement. This essential point will be futher elaborated upon later. In the case of algebraic multigrid, the arrows indicate that the coarse systems are generated from a fine grid system: $A_H = P^T A_h P$.

In the following, some key differences between geometric and algebraic multigrid will be highlighted.

Geometric multigrid

- **Uses information from discretization.** The properties of the grid are utilized in geometric multigrid. This is in the opinion of the author an advantage since it provides valuable information compared to only using the coefficients of the system matrix.

- **Nonlinear multigrid solvers (FAS-type)** can be formulated in geometric multigrid, since the access to the grid allows for the construction of coarse nonlinear operators.
- **Specialization to the problem** is often necessary in geometric multigrid to construct an efficient and robust solver. This specialization typically entails coarsening only in specific directions of the domain, where the coupling is weakest. Also the smoother is chosen based on the problem at hand.
- **Requires hiearchy of nested grids.** In connection to general unstructured grids, this is the biggest downside of geometric multigrid. For unstructured grids it may prove impossible to construct such a hiearchy of nested grids. At least if the initial unstructured grid has complex features and consists of many cells, it may prove impossible to construct a nested sequence of coarser grids. If the initial grid is coarse enough, the hierarchy of grids can be constructed by refinement.

Algebraic multigrid

- **No access to discretization.** Since the grid information is ignored in algebraic multigrid, this important information is lost.
- **Black box linear solver.** The fact that the method only requires the coefficients of the system matrix, allows for a black box solver with a minimal amount of information between various modules of the code. This enables a plug-n-play type modular code, where the solver can be switched out very easily.
- **Robustness.** The method is very robust since it only looks at the magnitude of the system coefficients and therefore automatically adapts to the problem at hand. As long as the problem is elliptic in nature and the matrix is symmetric positive definite or a M-matrix, it is likely to work.
- **No nonlinear multigrid solvers (FAS-type).** To the knowledge of the author, formulating a FAS-type multigrid solver is impossible in the context of classic algebraic multigrid. Instead, global linearization with e.g. Newton's method or Picard iterations is required before the application of algebraic multigrid.

Note that in the context of algebraic multigrid, the error when depicted as in Figure 1.1 may not appear smooth. Still the application of relaxation may not effectively remove this error. This type of error is denoted as algebraic smooth

error. In a similar manner as in geometric multigrid, the interpolation matrix P needs to be constructed such that algebraic smooth error is effectively removed.

The element-based Algebraic Multigrid (AMGe) introduced in Chapter 2 can not be categorized as either geometric or algebraic multigrid. It can be viewed as a mix and attempts to draw on the best qualities of both methods. In the following, a list of its properties is provided.

Element-based Algebraic Multigrid (AMGe)

- **Uses information from discretization.** AMGe uses the topology of the mesh and the finite element stiffness matrices.
- **Nonlinear multigrid solvers (FAS-type)** can be constructed.
- **Specialization to the problem** is often needed for AMGe but typically less so than in geometric multigrid due to the fact that AMGe more easily adapts to anisotropies and jumping coefficients. Note that this all depends on the version of AMGe.
- **Requires hiearchy of nested grids.** AMGe does not need a hierarchy of nested grids but rather works on the topology of the mesh. Specifically, the mesh relations such as element to element, element to face, etc. are used.
- **Black box linear solver.** Compared to AMG, AMGe is less of a black box linear solver. In addition to requiring system coefficients as AMG, AMGe also requires the passing of the mesh topology and access to the local finite element matrices. Depending on the version of AMGe and the type of finite elements, the method may also require some knowledge of the underlying equations and the choice of discretization method.
- **Robustness.** Once specialized to the problem at hand, AMGe is highly robust and typically it is able to solve problems not easily solved with AMG. Also compared to geometric multigrid, AMGe provides more accurate interpolation, which results in increased robustness.

1.5 Main contributions

The work in this thesis is application-oriented and puts emphasis on developing practical solutions to realistic problems. This is accomplished by contributing

to bridging the gap between advanced numerical techniques and practical application. The overarching goal of this research is to accelerate and improve the quality of porous media flow simulations.

The contributions of this work are listed in the following.

- We are the first to present the FAS method for three-phase flow in 3D with gravity, coupled well model and with a heterogeneous benchmark, extending the previous study of [86]. In this work, numerical experiments provide a fair comparison between the conventional and FAS multigrid techniques, as well as highlighting interesting properties of the FAS multigrid algorithm that show promise with respect to addressing some of the key challenges in reservoir simulation. Furthermore, it is demonstrated that by combining FAS with Newton's method in a hybrid strategy, the best of both methods can be obtained. To our knowledge, this work is the first to present nonlinear multilevel preconditioning for these equations. A study on the feasibility of the FAS method for reservoir simulation is particularly relevant as many-core hardware is starting to be adopted by the industry and supported by commercial software packages.
- The work described in this thesis introduces the application of one version of AMGe with guaranteed approximation properties, [73, 72, 99], to the incompressible two-phase flow equations for reservoir simulation. The framework is completely recursive, allowing for multilevel variational upscaling with guaranteed approximation properties. The work demonstrates multilevel upscaling for two challenging test cases. In addition to upscaling the mixed system for pressure and velocity, the saturation equations are also upscaled with the same coarse spaces used for the pressure. To our knowledge there is no other method that supports mixed finite element formulations on general unstructured grids; it allows for multilevel nested hierarchies as well as it allows for great flexibility in the construction of the coarse spaces - with two possible strategies to locally enrich the coarse spaces by either using finer agglomerates or adding additional degrees of freedom for each agglomerate. Some highlights are the ability to handle aggregates with non-planar faces, and achieve local mass conservation on all levels, and provide support for well models. The AMGe approach possesses all these properties with the additional flexibility to assign a variable number of degrees of freedom per agglomerated face (interface between two agglomerated elements) that is automatically determined by the desired accuracy and by the topology of the agglomerated face by means of SVD.
- The work on nonlinear multigrid is further extended by introducing FAS based on AMGe with guaranteed approximation properties. The method

is demonstrated on a simplistic but numerically challenging porous media flow model. The method is a first step towards an efficient solver for fully implicit variational upscaling.

- We are the first to demonstrate Multilevel Monte Carlo based on AMGe for the two-phase incompressible flow equations for reservoir simulation.
- Finally, we demonstrate that the same coarse spaces (interpolation operators) can be used for both linear solvers/preconditioners and variational upscaling. This is needed to be able to solve variationally upscaled systems, which despite being upscaled still can be very large.

1.6 Software

All the work in this thesis is based on implementing software solutions, which tests new ideas and methods. In the following, a list of the codes developed during the course of this thesis is provided. All software is developed in C++.

- A three-phase, oil, water and gas compressible fluids/rock, immiscible fluids finite volume reservoir simulator, which the author wrote together with Klaus Langgren Eskildsen has been improved and extended to include a coupled well model and saturation tables to model relative permeabilities. Furthermore, FAS was extended to accommodate the well model and a hybrid FAS/Newton's method was implemented.
- A two-phase oil and water incompressible mixed formulated reservoir simulator with gravity and simple well model has been implemented on top of a two-level AMGe code developed by Ilya Lashuk. The two-level AMGe code provided the mesh agglomeration and the associated interpolation operator matrix P. Based on this code, two solvers were implemented and compared. Namely a block diagonal preconditioner based on the Auxiliary Space AMG solver (ADS) and a multigrid preconditioner based on AMGe with a cell-based (or agglomerate-based) Vanka smoother have been implemented. The block diagonal preconditioner was accelerated by MINRES and the AMGe based multigrid preconditioner was accelerated by GMRES. The code was based on MFEM and Hypre from Lawrence Livermore National Laboratory.
- The same model as before was reimplemented based on a new implementation of AMGe primarily implemented by Umberto Villa. The AMGe software is called Elag (short for element agglomeration) and supports mesh agglomeration and the construction of a multilevel hierarchy of coarse

spaces for the full de Rham sequence. It is a sequential code again based on MFEM and Hypre. This new implementation of the reservoir simulator based on multilevel AMGe was used for studying its accuracy as a variational upscaling tool. Furthermore a number of preconditioners were implemented for this model. These include the block diagonal preconditioner L^2-H^1 (see details in Chapter 3) based on a Schur complement and an AMGe based multigrid solver using a Restricted Additive Overlapping Schwarz smoother.

- A distributed parallel implementation (MPI) of the mixed formulation of the reservoir simulation equations were implemented based on the MFEM code. A solver based on a parallel version of L^2-H^1 was implemented and the scalability of the overall code was studied.

- The mixed formulation of the reservoir simulation equations was reimplemented in a parallel (MPI) multilevel AMGe setting based on ParElag. ParElag is the distributed parallel (MPI) version of Elag and also builds on MFEM and Hypre. It supports mesh agglomeration in parallel and the construction of a multilevel hierarchy of coarse spaces for the full de Rham sequence in parallel. The implementation of the parallel reservoir simulator based on ParElag can be used for variational upscaling. It has been tested and works, however a full performance study is still needed. Some hurdles still needs to be overcome. For instance, when solving an upscaled problem, we rely on the Multilevel Divergence Free preconditioner (see Chapter 3), which in parallel do not yet have an efficient coarsest grid solver. Furthermore, in parallel, the coarsest grid problem can not currently be made smaller than 1 agglomerate per processor. This means that for many processors, the coarsest grid problem remains relatively large and obstructs the ideal performance of the scheme. This needs to be resolved before conducting a full performance study of the code.

- The code for variational upscaling of the mixed formulation of the reservoir simulation equations was interfaced with a Multilevel Monte Carlo code written by Umberto Villa.

- A Full Approximation Scheme (nonlinear multigrid) solver was implemented in parallel in the AMGe framework ParElag. To test this solver, a mixed (velocity/pressure) nonlinear model for steady-state porous media flow was implemented. Furthermore, inexact and exact versions of Newton's method and Picard iterations were implemented to compare with the FAS-AMGe solver.

- The in-house compositional (and black oil) reservoir simulator (COSI) was parallelized in a distributed setting using PETSc. The original COSI code constituted more than 100,000 lines of old Fortran code. Details of this

are given in Chapter 10. This task was accomplished together with Stefan Lemvig Glimberg.

Much of the software developed in this work uses the finite element library MFEM, [1], from Lawrence Livermore National Laboratory (LLNL). MFEM is a general, modular, parallel C++ library for finite element methods research and development. It supports a wide variety of finite element spaces in 2D and 3D, as well as many bilinear and linear forms defined on them. It includes classes for dealing with various types of triangular, quadrilateral, tetrahedral and hexahedral meshes and their global and local refinement. Parallelization in MFEM is based on MPI, and it leads to high scalability in the finite element assembly procedure. It supports several solvers from the Hypre library, [53].

1.7 Thesis outline

The thesis is structured as a paper-based thesis, where in addition to the papers, introductions to relevant subjects are provided and gaps are filled out to provide a coherent storyline for the work carried out during the thesis. Chapter 2 serves as a general introduction to the AMGe method used throughout a large part of the thesis. In this chapter, the techniques used for agglomeration of the mesh are described as well as for the construction of the coarse spaces (interpolation operators). Chapter 3 introduces a number of preconditioners, which have been used throughout the thesis work. These include block diagonal preconditioners based on Schur complements and the AMG or ADS solvers from the Hypre library as well as preconditioners based on the interpolation operators generated by the AMGe method. The preconditioners have either been used for the solution of the AMGe upscaled problems or as part of a finest grid solver. Some comparisons between the various preconditioners are provided.

Chapters 5-8 constitute the four papers. The first paper studies the Full Approximation Scheme for a standard finite volume formulated oil, gas and water simulator. The second paper investigates AMGe as a variational upscaling tool for a mixed formulation of the reservoir simulation equations. The third paper complements Paper II by showing that the same AMGe coarse spaces can be used for solving the upscaled problems efficiently. The fourth paper circles back to the Full Approximation Scheme and demonstrates how it can be combined with AMGe to provide an efficient nonlinear multigrid solver.

The papers are followed up by Chapter 9, where it is demonstrated how the variational upscaling of the reservoir simulation equations using AMGe can be

used for the acceleration of uncertainty quantification. In particular, it is demonstrated how Multilevel Monte Carlo based on AMGe can estimate the mean and variance of the water cut in a production well, where the permeability field is treated as a stochastic variable. Finally in Chapter 10, the developments in the in-house compositional (and black oil) reservoir simulator COSI are explained. In particular, it is described how the code has been parallelized using the PETSc framework. Furthermore, the new parallel solvers are introduced and a study of the parallel scalability is provided.

Chapter 2

Element-based Algebraic Multigrid (AMGe)

AMGe is a framework of multilevel methods for the solution of systems stemming from finite element discretizations. It was first introduced in [25], where interpolation operators based on multigrid convergence theory and finite element stiffness matrices were developed. Since then, numerous versions of AMGe has been developed. This chapter will introduce the specific version of AMGe used in this work. The focus will be on the practical aspects of the methods. We refer the reader to the referenced papers for the proofs behind the method. The commonality in all AMGe methods is that they construct interpolation operators from solving many local problems. This is a key feature, because it fits nicely with current hardware trends, where data locality is important. The method is an obvious candidate for a MPI-OpenMP hybrid parallelization strategy, where using MPI, each node on a cluster is given a part of the domain and inside each node, shared memory parallel threading is used to solve the local problems.

The original version of AMGe constructs an artificial grid based on the coefficients of the system matrix and then, on the basis of multigrid convergence theory, it introduces an interpolation operator constructed by using finite element stiffness matrices and by solving local problems, [25]. The goal of that work was to develop a more robust multigrid method for solving difficult problems for which AMG is struggling. This was done in the context of finite elements,

where the access to finite element stiffness matrices could be used. The usage of local stiffness matrices is one of the major differences between classical AMG and AMGe. In AMG, only the coefficients of the system matrix are used. This provides a big advantage in terms of having a black box solver, which can target linear systems coming from many different discretization schemes. However, by including more information about the discretization, it is possible to develop multilevel methods which are either more efficient than AMG or are able to solve more difficult problems than AMG can handle.

Another version of AMGe, leading to the specific one used in this work, eliminates the artificial grid and introduces coarse grids found by agglomeration of fine grid elements, [63]. That is, fine grid elements are partitioned into unions of elements called agglomerates. The coarse grids are then accompanied by compatible interpolation operators which provides coarse grid basis functions with a minimal energy property. These interpolation operators are local to the agglomerate of elements. The meaning of "compatible" interpolation is that interpolation to degrees of freedom shared by agglomerates is unique. That is, two neighbouring agglomerates sharing degrees of freedom will have the same interpolation on the shared degrees of freedom. This means that 1) finite element matrices for coarse elements are variationally related to the fine grid finite element matrices; and 2) the global coarse system matrix is variationally obtained (RAP Galerkin procedure) from the global fine grid system matrix. The method is split in two stages: In the first stage, the agglomerates are found and the associated coarse grid is formed. In the second stage, local interpolators from coarse to fine degrees of freedom are computed.

The two versions of AMGe referenced above are for nodal finite elements. The version of AMGe described in the following extends the method to the full de Rham sequence, [99, 72, 73]. That is the sequence covering H^1-conforming, $H(\mathrm{curl})$-conforming, $H(\mathrm{div})$-conforming and L^2-conforming finite element spaces. Specifically, the method is developed for the piecewise linear H^1-conforming, lowest order Nédélec, lowest order Raviart–Thomas and piecewise discontinuous constant finite elements.These finite element spaces are used in the solution to many different problems where curl, div and grad differential operators are in play. These include porous media flow, elliptic PDEs, Brinkman's problem and Maxwell's equations. The de Rham sequence can be written as

$$\mathbb{R} \hookrightarrow \widetilde{S}_h \xrightarrow{\nabla} \widetilde{Q}_h \xrightarrow{\nabla\times} \widetilde{\mathcal{R}}_h \xrightarrow{\nabla\cdot} \widetilde{\mathcal{W}}_h, \tag{2.1}$$

where for the purposes of the description of the method in the following

- $\widetilde{S}_h$ is the space of continuous linear functions
- $\widetilde{Q}_h$ is the Nédélec space of lowest order

- $\widetilde{\mathcal{R}}_h$ is the lowest-order Raviart-Thomas space
- $\widetilde{\mathcal{W}}_h$ is the space of constant functions (piecewise discontinuous).

It is an exact sequence, when the domain Ω is homeomorphic to a ball, [87]. The term "exact" means in this context, [15, 14], that

(1) $\quad \nabla\widetilde{S}_h \subset \widetilde{Q}_h,\ \nabla\times\widetilde{Q}_h \subset \widetilde{\mathcal{R}}_h,\ \nabla\cdot\widetilde{\mathcal{R}}_h \subseteq \widetilde{\mathcal{W}}_h$

(2) $\quad \ker(\nabla) = \mathbb{R},\ \ker(\nabla\times) = \nabla\widetilde{S}_h,\ \ker(\nabla\cdot) = \nabla\times\widetilde{Q}_h,\ \widetilde{\mathcal{W}}_h = \nabla\cdot\widetilde{\mathcal{R}}_h.$

Figure 2.1 illustrates the associated finite elements.

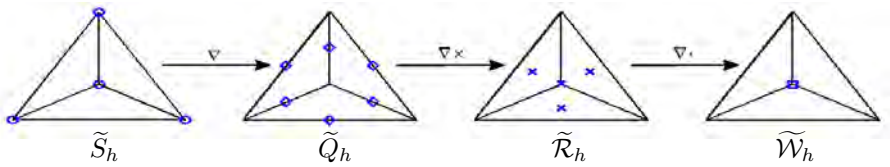


Figure 2.1: Illustration of the lowest-order finite elements associated with the de Rham sequence. The circles represent the degrees of freedom associated with the vertices. The diamonds represent the degrees of freedom associated with the edges. The crosses represents the degrees of freedom associated with the faces. The squares represent the degrees of freedom associated with the element.

The spaces S_h, Q_h $\mathcal{R}_h$ and $\mathcal{W}_h$ (without tildes) are those with zero boundary conditions on $\partial\Omega$. That is $S_h \subset \widetilde{S}_h$ are defined to consist of functions from $\widetilde{S}_h$, which vanish on $\partial\Omega$. $Q_h \subset \widetilde{Q}_h$ consists of functions which have zero tangential component on $\partial\Omega$. $\mathcal{R}_h \subset \widetilde{\mathcal{R}}_h$ consists of functions with zero normal component on $\partial\Omega$ and $\mathcal{W}_h \in \widetilde{\mathcal{W}}_h$ is defined to consist of functions which have zero average over Ω, [72].

The initial approach to the method was first described in [99]. Here a version of AMGe was introduced, which generates coarse basis functions on macro-elements (agglomerates) with the property that the resulting spaces are subspaces of the original de Rham sequence. Given that a number of topological requirements for the agglomerates are met, the method provides coarse spaces which form an exact de Rham sequence. That is (1) and (2) are satisfied. In addition, if coarse faces are flat and coarse edges are straight, the H^1-conforming coarse space interpolates exactly affine functions on each agglomerate and the $H(\mathrm{curl})$-conforming and $H(\mathrm{div})$-conforming coarse spaces interpolate exactly vector constants on each agglomerate. For general unstructured meshes, those approximation properties are not enforced.

The method was further extended for the lowest order Raviart-Thomas finite element space in [73] by introducing coarse spaces with improved/guaranteed approximation properties. This is ensured by introducing additional degrees of freedom associated with non-planar faces between agglomerates. In this way, on each agglomerate the coarse spaces interpolates exactly all vector constants. The initial approach to the method as explained in [99] only have these approximation properties if the coarse faces are flat. Similarly, the method in [99] also did not have these approximation properties for $H(\text{curl})$-conforming coarse spaces if the coarse edges were not straight. This was finally remedied in [72] where the full de Rham sequence with improved/guaranteed approximation properties was completed by also introducing an $H(\text{curl})$-conforming coarse spaces and H^1-conforming coarse space with improved/guaranteed approximation properties. It should be noted that the version of AMGe with guaranteed approximation properties results in denser linear systems on the agglomerated mesh.

The remainder of this chapter is dedicated to explaining the particular method used in this work. Following an introduction to mesh agglomeration techniques, the explanation will initially follow that in [99], where the construction of coarse spaces forming an exact de Rham sequence is introduced. Following that, the extension to coarse Raviart-Thomas spaces with improved/guaranteed approximation properties for the lowest order Raviart-Thomas space is explained, [73]. As mentioned previously, the method in [99] only have improved/guaranteed approximation properties for the coarse Raviart-Thomas space, when the coarse faces are flat. Since in this work, improved/guaranteed approximation properties have not been used for the coarse $H(\text{curl})$ space, the method will not be treated here and we refer the reader to [72] for the description. The reason for this is that the coarse $H(\text{curl})$ space have only been used for smoothing purposes in a multigrid solver introduced in section 3.2.2 and therefore there was no need for such accurate (and heavy) interpolation operators in this context. The coarse H^1 space have not been used at all in this work, but for the sake of completeness of the de Rham sequence, it is still mentioned briefly. Note that the AMGe code (Elag and ParElag) used throughout this thesis work is implemented such that it easily allows to turn on or off these approximation properties.

2.1 Mesh agglomeration

Before describing how agglomerated meshes can be formed, we need to introduce the storage format being used for the mesh relations. We use the well-known CSR format to hold the mesh relations. This is best explained by an

example. For instance, `element_element` is a sparse matrix without the value array, but where the location of the non-zeros in this case provides the element-to-element relations. These sparse matrices are here called boolean matrices. `element_element` is a sparse boolean matrix with as many rows and columns as there are fine grid elements. If there is a non-zero in location i, j, this means element i and element j are neighbours in the mesh.

We encode a mesh on any grid level by the following connectivity tables (represented as CSR matrices):

- `element_face`
- `face_edge`
- `edge_vertex`

The entities `element`, `face`, `edge` and `vertex` may refer to either a fine grid entity or an agglomerated entity. This depends on the given level.

To enable the connectivity tables above to be formed recursively on coarser grid levels, we construct the `AEntity_entity` tables:

- Agglomerated element to element: `AE_element`
- Agglomerated face to face: `AF_face`
- Agglomerated edge to edge: `AEdge_edge`
- Agglomerated vertex to vertex: `AV_vertex`

The purpose of the `AEntity_entity` tables are to describe which finer grid entities belong to which agglomerated entity. In the following, it is described how to compute the connectivity tables and the `AEntity_entity` tables for the various entities.

2.1.1 Agglomerated elements

To construct agglomerated elements using graph partitioning techniques, we build the dual graph of the mesh, which is an undirected graph, where each node of the graph represents an element in the mesh and node i is connected to node j if element i and element j share a face. METIS, [66], is used for

the partitioning of the undirected graph resulting in agglomerates consisting of fine-grid elements. Weights of the nodes and links in the dual graph (i.e. for the elements and faces of the mesh) can be provided to the partitioners in order to generate smaller agglomerated elements in parts of the domain or to modify the aspect ratio of the coarse elements. In practice, the element agglomeration is done by constructing the CSR formatted sparse matrix `element_element`. The sparse boolean matrix `element_element` is then given to METIS along with a request for the number of partitions (the desired number of agglomerates). METIS then returns an integer array with the same size as the number of fine grid elements. Each entry of that array corresponds to a particular fine grid element. The entries are filled out by the values: $0, 1, \ldots, \mathrm{nAE} - 1$, which indicate the specific partition each fine grid element belongs to. Here AE stands for Agglomerated Element. nAE is the number of agglomerated elements. Based on the partitioning array, the boolean sparse matrix: `AE_element` can be setup in a straightforward manner. `AE_element` describes which fine grid elements belong to which agglomerated element. Here AE stands for Agglomerated Element. `AE_element` has as many rows as there are agglomerates and as many columns as there are finest grid elements. The non-zero structure (or sparsity pattern) of `AE_element` is such that there exists a non-zero in location i, j, when agglomerate i contains fine grid element j.

This procedure can be made in a recursive fashion such that agglomerated elements are further agglomerated into even bigger parts. This can be done by providing the agglomerated element to agglomerated element matrix: `element_element`[1], where [1] refers to the grid level (grid level [0] being the finest) to METIS. It can be computed via the following boolean sparse Matrix-Matrix-Matrix product

$$\texttt{element\_element}[1] = \texttt{AE\_element}[0]\ \texttt{x}\ \texttt{element\_element}[0]\ \texttt{x}\ \texttt{AE\_element}^{\mathrm{T}}[0]$$

For more details on linear algebra operations on mesh relation matrices, see [114].

2.1.2 Agglomerated faces

A coarse face consists of a union of fine faces on the boundary of two neighbouring agglomerated elements (or a single agglomerate if it is on the boundary of the computational domain). The relation between the agglomerated elements and the faces can be computed via the following boolean sparse matrix-matrix product

$$\texttt{AE\_face} = \texttt{AE\_element}\ \texttt{x}\ \texttt{element\_face}. \tag{2.2}$$

The agglomerated face to face relations: `AF_face` can be found by computing

minimal intersection sets on the following sparse matrix

$$\texttt{face\_AE\_face} = \texttt{AE\_face}^{\texttt{T}}\ \texttt{x}\ \texttt{AE\_face}. \tag{2.3}$$

Note that contrary to before, the value array in the CSR format is used in this context. Entry i, j in `face_AE_face` provides the number of agglomerated elements shared by face i and face j. This can be $0, 1$ or 2. Minimal intersection sets are found by looking at each row i in `face_AE_face` and identify where entry $\texttt{face\_AE\_face}(i, j) = \texttt{face\_AE\_face}(i, i)$. If this is the case, face i and face j belong to the same coarse face. For more details on this approach, see [114].

2.1.3 Agglomerated edges

The coarse edges are found by considering all the fine edges that are boundary edges of a coarse face. This set of fine edges is then partitioned into non-intersecting subsets called coarse edges. Similarly to before, the agglomerated face to edge relations can be computed as

$$\texttt{AF\_edge} = \texttt{AF\_face}\ \texttt{x}\ \texttt{face\_edge}. \tag{2.4}$$

The agglomerated edge to edge relations: `AEdge_edge` can be found by computing minimal intersection sets on the sparse matrix

$$\texttt{edge\_AF\_edge} = \texttt{AF\_edge}^{\texttt{T}}\ \texttt{x}\ \texttt{AF\_edge}. \tag{2.5}$$

Computing minimal intersection sets can be done in a similar way to what is done for coarse faces.

2.1.4 Agglomerated vertices

The coarse vertices are defined as the endpoints of the coarse edges. Similarly to before, the agglomerated edge to vertex relations can be computed as

$$\texttt{AEdge\_vertex} = \texttt{AEdge\_edge}\ \texttt{x}\ \texttt{edge\_vertex}. \tag{2.6}$$

The agglomerated vertex to vertex relations: `AV_vertex` can be found by computing minimal intersection sets on the sparse matrix

$$\texttt{vertex\_AEdge\_vertex} = \texttt{AEdge\_vertex}^{\texttt{T}}\ \texttt{x}\ \texttt{AEdge\_vertex}. \tag{2.7}$$

The result of applying this type of agglomeration procedure is illustrated in Figure 2.2.

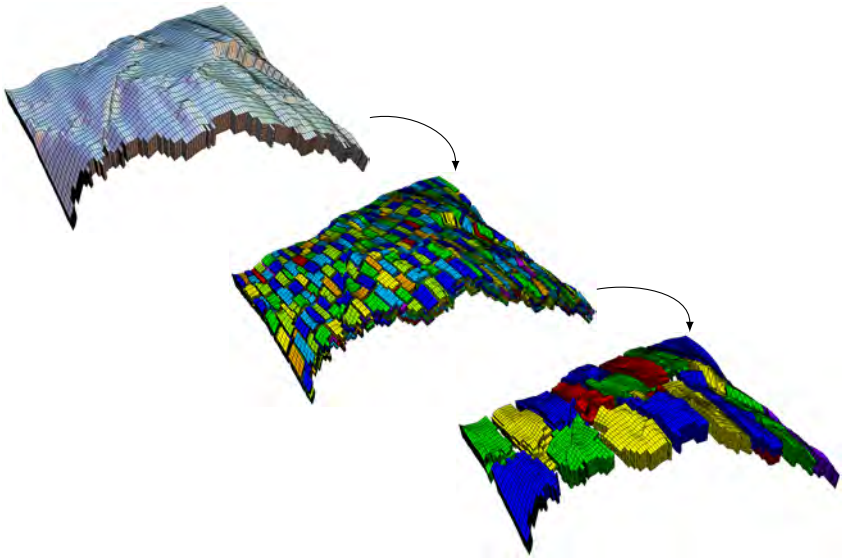


Figure 2.2: Illustration of agglomeration outcome.

As part of the thesis work, the agglomeration framework was expanded to allow selected elements to remain unagglomerated. This was done to facilitate certain needs in relation to well models in reservoir simulation. In particular, it was found by experimentation (Paper II) that letting elements containing wells (and possibly their nearest neighbours) remain unagglomerated, significantly improved accuracy of the upscaled discretizations.

2.1.5 Agglomeration in parallel

In the current implementation, the agglomeration takes place locally on each processor. First the domain is split into one subdomain per processor and then the agglomeration is performed locally on each subdomain. Figure 2.3 illustrates this using 48 processors.

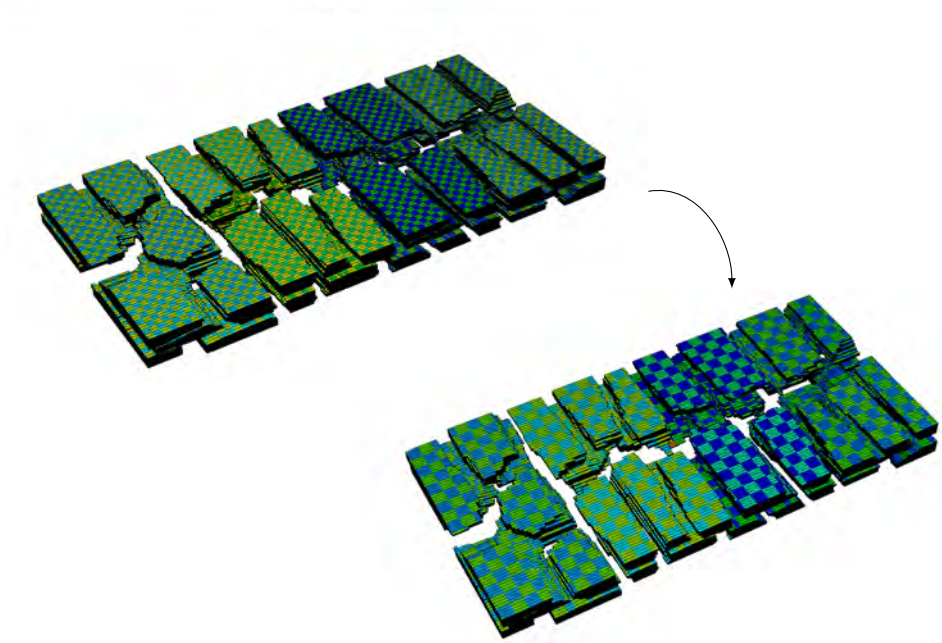


Figure 2.3: Illustration of agglomeration in parallel on 48 processors. Here level 1 and level 2 are displayed. Level 0 is the finest grid and level 2 is the coarsest grid.

The downside of this approach is that the coarsest problem will have as many agglomerated elements as there are processors. Therefore, when using many processors, the coarsest grid problem can easily be too large. In the context of multigrid solvers, it would typically be required to have a smaller coarsest grid problem. This issue could be alleviated in a future version of the code by either (or both)

- dropping processors and relocating data between processors on the way down in the hierarchy.
- letting agglomerates span across several processors.

2.1.6 Topological constraints for the agglomeration

A number of topological requirements for the agglomerated meshes are necessary to allow for the construction of coarse spaces. These topological requirements

are necessary to formulate meaningful and solvable local problems in the construction of the interpolation operators. As an example, agglomerated elements can not form donuts or have holes. If this was the case, the local problem associated with computing coarse basis functions for e.g. $H(\mathrm{curl})$ spaces would not have a unique solution. In the following, a number of topological requirements are listed for 3D agglomerated meshes

- *Requirements for agglomerated elements:*
  1. Fine elements belonging to an agglomerated element need to form a single connected component (two elements are connected if the share a face).
  2. Fine faces on the boundary of an agglomerated element need to form a single connected component (two fine faces on the boundary of an agglomerated element are connected if they share an edge which belongs to the boundary of the agglomerated element).
  3. Each fine edge on the boundary of the agglomerated element is shared by exactly two fine faces, which themselves belong to the boundary of the agglomerated element.
  4. The face-patch of each fine vertex on the boundary of the agglomerated element (i.e. the set of all the faces on the boundary of the agglomerated element that touch the vertex) needs to be a single connected component (two boundary faces are connected if the share a boundary edge).
- *Requirements for agglomerated faces:*
  1. Fine faces belonging to an agglomerated face need to form a single connected component (two faces are connected if the share an edge).
  2. Fine edges on the boundary of an agglomerated face need to form a single connected component (two fine edges on the boundary of an agglomerated face are connected if they share a vertex which belongs to the boundary of the agglomerated face).
  3. Each fine vertex on the boundary of the agglomerated face is shared by exactly two fine edges that belongs to the boundary of the agglomerated face.
- *Requirements for agglomerated edges:*
  1. The fine edges in an agglomerate edge need to form a single connected component.
  2. Two agglomerated edges can not share both end points.

In serial, if an agglomerated entity is bad, the agglomerated entity is de-agglomerated and a greedy algorithm was used to re-agglomerate into two or more agglomerated entities.

In the current and parallel (MPI) implementation, bad agglomerated entities are de-agglomerated and (for now) left unagglomerated until the next coarser grid level, where they are subject to agglomeration again. In the parallel implementation, a greedy algorithm is currently not being used to re-agglomerate bad agglomerates, since the implementation would be non-trivial and require communication between MPI-processes. In the opinion of the author, it is important to investigate techniques capable of re-agglomerating bad agglomerates. By experimentation it was found that if too many agglomerated entities are deemed bad and hence de-agglomerated, it is difficult to maintain good coarsening factors (in terms of reduction in degrees of freedom and non-zeros for the coarse systems). This may easily ruin the overall performance of the multilevel method. More aggressive coarsening (including more fine entities in an agglomerate) can be used to counteract this, however this comes with the downside that the local problems solved in the construction of the coarse spaces become larger and computationally more expensive.

Note that depending on the application of AMGe, it is not necessarily needed to construct the full sequence of agglomerated entities. For instance, if only the coarse Raviart-Thomas space are needed, it is only necessary to form agglomerated elements and agglomerated faces. Similarly, if coarse Raviart-Thomas and coarse Nédélec spaces are needed, the construction of agglomerated elements, faces and edges are required, but agglomerated vertices are unnecessary.

Evidently, the construction of agglomerated meshes for general unstructured meshes is a non-trivial process. If only the coarse Raviart-Thomas spaces are of interest, the topological requirements are much easier to meet. However, when coarse Nédélec or coarse H^1 spaces are needed, forming agglomerated meshes is quite difficult for general unstructured meshes. This is still an ongoing research topic, which, in the opinion of the author, require significant effort to resolve.

2.2 Coarse $H(\mathbf{div}) - L^2$ spaces (Raviart-Thomas)

The coarse L^2 spaces are constructed in the same way as introduced in [99], where the coarse space consists of piecewise constant functions on agglomerated elements. We define the coarse L^2 space $\widetilde{\mathcal{W}}_H \subset \widetilde{\mathcal{W}}_h$ to consist of functions which are constant on each agglomerated element. A basis of the space $\widetilde{\mathcal{W}}_H$ is then used to form the columns of the prolongation matrix $P : \widetilde{\mathcal{W}}_H \to \widetilde{\mathcal{W}}_h$. In this

section, we will demonstrate how to construct a coarse $H(\mathrm{div})$ space $\widetilde{\mathcal{R}}_H \subset \widetilde{\mathcal{R}}_h$ such that $\nabla \cdot \widetilde{\mathcal{R}}_H = \widetilde{\mathcal{W}}_H$. This property is necessary to preserve the stability of the upscaled discretization and to guarantee that the spaces $(\widetilde{\mathcal{R}}_H, \widetilde{\mathcal{W}}_H)$ are inf-sup compatible. First for each coarse element T we define the local finite element spaces:

$$\mathcal{R}_T = \left\{ \mathbf{v}_h \in \widetilde{\mathcal{R}}_h \,|\, \mathrm{supp}(\mathbf{v}_h) \subset T \,\text{and}\, \mathbf{v}_h \cdot \mathbf{n} = 0 \,\text{on}\, \partial T \right\},$$
$$\text{and} \quad \mathcal{W}_T = \left\{ w_h \in \widetilde{\mathcal{W}}_h \,|\, \mathrm{supp}(w_h) \subset T \,\text{and}\, (w_h, 1) = 0 \right\}.$$

The basis function associated with a coarse face is set agglomerated element by agglomerated element. First, on the agglomerated element T, it is defined by setting its boundary normal components equal to 1 on the triangles of the coarse face (given it is tetrahedrons) and 0 on the triangles of the remaining faces of ∂T. Then the following mixed problem is solved to define the basis function in the interior of the neighbouring agglomerated elements.

PROBLEM 1 *Find* $(\mathring{\mathbf{r}}_T, p_h) \in \mathcal{R}_T \times \mathcal{W}_T$ *such that*

$$\begin{cases} ((\mathring{\mathbf{r}}_T + \mathbf{r}_F), \mathbf{v}_h)_T + (p_h, \nabla \cdot \mathbf{v}_h)_T = 0, & \forall\ \mathbf{v}_h \in \mathcal{R}_T \\ (\nabla \cdot (\mathring{\mathbf{r}}_T + \mathbf{r}_F), w_h)_T = 0, & \forall\ w_h \in \mathcal{W}_T \end{cases}$$

where $\mathbf{r}_F$ is a function with vanishing normal traces on ∂T, except for the coarse face $F \in \partial T$, where $\mathbf{r}_F \cdot \mathbf{n}_F = 1$. Here $\mathbf{n}_F$ is defined as follows: for each coarse face F, we associate a unit normal vector $\mathbf{n}_F$. The orientation of $\mathbf{n}_F$ is arbitrarily chosen and fixed to point outside one of the two agglomerated elements, which share the coarse face F. The same convention holds for $\mathbf{n}_f$, i.e., it is a unit normal vector associated with each fine grid face f and $\mathbf{n}_f$ has an arbitrary, but fixed, chosen direction. Let $f = 1$ or $f = -1$ depending on whether $\mathbf{n}_f$ and $\mathbf{n}_F$ have the same or opposite directions.

Problem 1 is guaranteed to have a unique solution, [73, 99]. By solving these local problems on each pair of agglomerates (T^+, T^-) adjacent to a coarse face F, we obtain the coarse basis functions $\mathbf{r}_h = \mathbf{r}_F + \mathring{\mathbf{r}}_{T^+} + \mathring{\mathbf{r}}_{T^-}$ of the space $\mathcal{R}_H$. We finally let the columns of the prolongation matrix $P : \widetilde{\mathcal{R}}_H \to \widetilde{\mathcal{R}}_h$ be the collection of the coarse basis functions $\mathbf{r}_h$.

2.3 Coarse $H(\mathbf{curl})$ spaces (Nédélec)

The methodology for the construction of the coarse Nédélec space is similar to that of the coarse Raviart-Thomas space, however there are some significant differences. The major difference is that two rounds of solving mixed problems are required. The first being the extension from the coarse edges into the interior of the neighbouring coarse faces. Next is the extension of the coarse basis functions into the interior of the neighbouring agglomerated elements.

Before addressing the extensions, we define the spaces

$$Q_T = \left\{ \mathbf{q}_h \in \widetilde{Q}_h \,|\, \mathrm{supp}(\mathbf{v}_h) \subset T \text{ and } \mathbf{q}_h \times \mathbf{n} = 0 \,\mathrm{on}\, \partial T \right\},$$
$$\text{and} \quad Q_F = \left\{ \mathbf{q}_h \in \widetilde{Q}_h \,|\, \mathrm{supp}(\mathbf{v}_h) \subset F \text{ and } \mathbf{q}_h \times \mathbf{n} = 0 \,\mathrm{on}\, \partial F \right\}.$$

Now the first extension is performed by solving the following local mixed problem for each coarse face F.

PROBLEM 2 *Find* $(\mathring{\mathbf{q}}, \mathbf{z}) \in Q_F \times \widetilde{\mathcal{R}}_F$ *such that*

$$\begin{cases} ((\mathring{\mathbf{q}} + \mathbf{q}_{\mathrm{edge}}), \mathbf{p})_F + (\nabla_F \times \mathbf{p}, \mathbf{z})_F = 0, & \forall\, \mathbf{p} \in Q_F \\ (\nabla_F \times (\mathring{\mathbf{q}} + \mathbf{q}_{\mathrm{edge}}), \mathbf{s})_F = 0 & \forall\, \mathbf{s} \in \widetilde{\mathcal{R}}_F, \end{cases}$$

where $\mathbf{q}_{\mathrm{edge}}$ is the function with vanishing tangential trace on ∂F, except for the agglomerated edge E where $\mathbf{r} \cdot \boldsymbol{\tau}_E = 1$. Here $\boldsymbol{\tau}_E = \pm \boldsymbol{\tau}_e$ for all fine edges e in a coarse edge E, $\boldsymbol{\tau}_e$ is the direction of the fine edge e and we take $\pm$ so that all the $\boldsymbol{\tau}_e$ point in the same direction.

Once the extension from coarse edges into the interior of the neighbouring coarse faces (Problem 2) is completed, the extension of the coarse basis functions proceeds by extending them into the interior of the neighbouring agglomerated elements. Again this is accomplished by solving local saddle point problems (Problem 3).

PROBLEM 3 *Find* $(\mathring{\mathbf{q}}_T, \mathbf{r}_T) \in Q_T \times \widetilde{\mathcal{R}}_T$ *such that*

$$\begin{cases} \left((\mathring{\mathbf{q}}_T + \mathbf{q}_{\mathrm{patch}(E)}), \mathbf{p}\right)_T - (\nabla \times \mathbf{p}, \mathbf{r}_T)_T = 0, & \forall\, \mathbf{p} \in Q_T \\ \left(\nabla \times (\mathring{\mathbf{q}}_T + \mathbf{q}_{\mathrm{patch}(E)}), \mathbf{s}\right)_T + (\nabla \cdot \mathbf{r}_T, \nabla \cdot \mathbf{s})_T \,, = 0 & \forall\, \mathbf{s} \in \widetilde{\mathcal{R}}_T \end{cases}$$

where patch(E) refers to the two agglomerated faces that share the agglomerated edge E. That is $\mathbf{q}_{\text{patch}(E)}$ is the extension of $\mathbf{q}_{\text{edge}}$ to the patch of E as computed by Problem 2. This finalizes the construction of the basis functions for the coarse Nédélec space. We let the columns of the prolongation matrix $P : Q_H \to Q_h$ be the collection of the coarse basis functions defined by Problem 3. We refer to [72] for the version with improved approximation properties.

2.4 Coarse H^1 spaces

Since the coarse H^1 space is not used in this thesis, it is only briefly explained in words here. The construction of the coarse H^1 space on coarse edges is done in the same way as the face-to-interior extension of the coarse $H(\text{div})$ space. The construction of the coarse H^1 space on coarse faces is done in the same way as the face-to-interior extension of $H(\text{curl})$. Finally, the construction of the coarse H^1 space on coarse elements is done by solving the Poisson equation with prescribed Dirichlet boundary conditions. We refer to [99] for the full description and to [72] for the version with improved approximation properties.

2.5 Coarse $H(\text{div}) - L^2$ spaces (Raviart-Thomas) with improved approximation properties

The method used to create coarse finite element spaces for the lowest order Raviart-Thomas space with improved/guaranteed approximation properties is described here. The explanation closely follows that of [73]. The method is a two-step process, where we first find the coarse basis functions on coarse faces and then extend the basis functions into the interior of the neighbouring agglomerated elements.

The coarse basis functions are defined in terms of their fine degrees of freedom. Given a sufficiently smooth vector function $\mathbf{r}$ and a fine face f, the value of the degree of freedom associated with f is defined as

$$\text{DoF}_f(\mathbf{r}) = \int_f \mathbf{r} \cdot \mathbf{n}_f \, dA, \tag{2.8}$$

where $\mathbf{n}_f$ is the unit normal to the fine face and A is the surface area of the face.

For each coarse face F, a matrix $\mathbf{W}_F$ is formed. $\mathbf{W}_F$ consists of the values of the DoF of the fine faces $f_1, \ldots, f_{|F|}$ constituting F

$$\mathbf{W}_F = \begin{bmatrix} \mathrm{DoF}_{f_1}(\mathbf{e}_1) & \mathrm{DoF}_{f_1}(\mathbf{e}_2) & \mathrm{DoF}_{f_1}(\mathbf{e}_3) & \mathrm{sgn}(f_1, F) \int\limits_{f_1} dA \\ \vdots & \vdots & \vdots & \vdots \\ \mathrm{DoF}_{f_{|F|}}(\mathbf{e}_1) & \mathrm{DoF}_{f_{|F|}}(\mathbf{e}_2) & \mathrm{DoF}_{f_{|F|}}(\mathbf{e}_3) & \mathrm{sgn}(f_{|F|}, F) \int\limits_{f_{|F|}} dA \end{bmatrix}, \tag{2.9}$$

where $|F|$ is the number of fine faces in the coarse face F. Here $\mathrm{sgn}(f, F) = 1$ if the orientation of the fine face is equal to that of the coarse face and $\mathrm{sgn}(f, F) = -1$ otherwise. Above, $\mathbf{e}_i$ stand for the three coordinate constant vector-functions. The goal is to ensure that the coarse Raviart-Thomas space contains locally (on each agglomerated element) these constant vectors, hence have first order of approximation in L^2 as the fine-grid Raviart-Thomas space. The above construction is fairly general; we can include any given functions in $H(\mathrm{div})$ of our interest in the coarse $H(\mathrm{div})$ space and maintain the compatibility Note that if

$$\mathbf{W}_F = \begin{bmatrix} \mathrm{sgn}(f_1, F) \int\limits_{f_1} dA \\ \vdots \\ \mathrm{sgn}(f_{|F|}, F) \int\limits_{f_{|F|}} dA \end{bmatrix}, \tag{2.10}$$

the method from [99] (as explained previously) is recovered. Using an SVD decomposition $\mathbf{W}_F = \mathbf{U}\Sigma\mathbf{V}^T$, the linearly dependent columns of $\mathbf{W}_F$ are eliminated. It is worth noticing that the cost of computing such SVD scales linearly with the number of fine degree of freedom which belongs to the coarse face. This is due to the fact that (1), we perform the *thin* SVD [48] and (2), that the number of columns in $\mathbf{W}_F$ is small and independent of the number of fine degree of freedom. The left singular vectors (columns of $\mathbf{U}$) $\mathbf{u}_j$ are chosen based on the corresponding singular values σ_j. If $\sigma_j \geq \epsilon\sigma_{\max}$, where $\epsilon \in (0, 1]$ is a user-given input, then $\mathbf{u}_j$ defines a coarse basis function for the coarse face F denoted $\mathbf{r}_F^j$. More precisely, $\mathbf{r}_F^j$ is only equal to $\mathbf{u}_j$ on the coarse face F and zero everywhere else. If the coarse face F is planar, then only σ_1 would be different from 0 and only one coarse trace will be selected, for a non-planar face up to four coarse traces will be selected depending on the tolerance ϵ.

The above procedure describes the first step to finding a coarse $H(\mathrm{div})$ space. The second step involves taking the partially defined functions $\mathbf{r}_F^j$ and extending these coarse basis functions into the interior of the coarse elements. The extension is performed using the approach in [99] (also described previously), which guarantees that the divergence of the coarse Raviart-Thomas space belongs to the coarse L^2 space.

Given a partially defined function $\mathbf{r}_F^j$ on the coarse face F belonging to the coarse element T, the local (element-based) mixed system reads

PROBLEM 4 *Find* $(\mathring{\mathbf{r}}_T, p_h) \in \mathcal{R}_T \times \mathcal{W}_T$ *such that*

$$\begin{cases} \left(\alpha\,(\mathring{\mathbf{r}}_T + \mathbf{r}_F^j), \mathbf{v}_h\right)_T + (p_h, \nabla \cdot \mathbf{v}_h)_T = 0, & \forall\ \mathbf{v}_h \in \mathcal{R}_T \\ \left(\nabla \cdot (\mathring{\mathbf{r}}_T + \mathbf{r}_F^j), w_h\right)_T = 0, & \forall\ w_h \in \mathcal{W}_T \end{cases}$$

The coefficient matrix α (a 3×3 SPD matrix) can be set equal to the coefficients from the original problem (such as permeabilities), but this is not strictly necessary. Problem 4 is guaranteed to have a unique solution, [73, 99]. By solving these local problems on each pair of agglomerates (T^+, T^-) adjacent to a coarse face F, we obtain the coarse basis functions $\mathbf{r}_h = \mathbf{r}_F^j + \mathring{\mathbf{r}}_{T^+} + \mathring{\mathbf{r}}_{T^-}$ of the space $\mathcal{R}_H$. We finally let the columns of the prolongation matrix $P : \widetilde{\mathcal{R}}_H \to \widetilde{\mathcal{R}}_h$ be the collection of the coarse basis functions $\mathbf{r}_h$.

2.6 Other versions of AMGe

The versions of AMGe introduced here only constitute a small selection of the body of work in this field. One of the more interesting directions of AMGe is the so called Spectral AMGe [27, 28, 65]. Here small eigenvalue problems are solved in order to construct interpolation operators. This is computationally expensive (especially for large agglomerates), but it provides the opportunity for very accurate interpolation operators. One of advantages of the Spectral AMGe method is the ability to adaptively choose the number of coarse basis functions per coarse grid entity. This means, in regions of the domain where more coarse grid basis functions are needed for satisfactory interpolation accuracy, these can be included. Similarly in regions of the domain, where the solutions is smooth, few coarse grid basis functions are needed. The difficulty in the Spectral AMGe method is how to choose the optimal number of coarse basis functions to include. From the understanding of the author, this is still an unresolved issue and one of the more difficult parts of the Spectral AMGe method. The optimal number of coarse basis functions needs to strike a balance between computational efforts and the accuracy of the interpolation operators.

Chapter 3

Multigrid preconditioners for mixed systems

The system of interest is the following mixed system

$$\begin{bmatrix} M & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix} = \begin{bmatrix} \mathbf{F_u} \\ F_p \end{bmatrix}, \tag{3.1}$$

where C may be a zero matrix or a diagonal matrix. M is the velocity mass matrix from a Raviart-Thomas discretization and B represents the mixed $H(\mathrm{div}) - L^2$ terms. The equations and the discretization used in reservoir simulation resulting in this system are addressed in more detail in Paper II.

A number of preconditioners have been used throughout the work described in this thesis. This chapter serves to summarize these preconditioners and comment on their individual strengths and weaknesses. Some of the preconditioners are based on AMGe coarse spaces for the interpolation between grid levels, where others are based on classical AMG or solvers found in the open-source library Hypre. The preconditioners can be classified as either

- Block diagonal preconditioners (based on a Schur complement or an augmented Lagrangian formulation).
- Fully coupled AMGe based preconditioners.

3.1 Block diagonal preconditioners

Two different block diagonal preconditioners have been implemented. These are here called:

- The $L^2 - H^1$ preconditioner based on a Schur complement and AMG.
- The $H(\mathrm{div}) - L^2$ preconditioner based on an augmented Lagrangian formulation and the Auxiliary Space AMG solver for $H(\mathrm{div})$ problems (ADS).

3.1.1 $L^2 - H^1$ preconditioner

An optimal block diagonal preconditioner for the saddle point system can be derived as follows. For any symmetric positive definite matrix H, we let $\widetilde{\Sigma} = BH^{-1}B^T + C$ and we define

$$\widetilde{\mathbf{A}} = \begin{bmatrix} H & B^T \\ B & -C \end{bmatrix}, \quad \text{and} \quad \mathbf{P} = \begin{bmatrix} H & 0 \\ 0 & \widetilde{\Sigma} \end{bmatrix}. \tag{3.2}$$

It is well-known that $\mathbf{P}$ is an optimal preconditioner for $\widetilde{\mathbf{A}}$, indeed in exact arithmetic preconditioned MINRES converges in 3 iterations [90]. By letting $H = \mathrm{diag}(M)$ and exploiting the fact that the finite element Raviart-Thomas mass matrix M is spectrally equivalent to its diagonal, it is possible to show that $\mathbf{P}$ is optimal also for the original problem $\mathbf{A}$. Moreover, since H is diagonal, the Schur Complement $\widetilde{\Sigma}$ is explicitly available and sparse, which allows us to approximate its inverse by a well-suited algebraic multigrid (AMG) solver. Specifically, we use BoomerAMG from Hypre [53]. The two diagonal blocks H and $\widetilde{\Sigma}$ of the preconditioner $\mathbf{P}$ in (7.8) are spectrally equivalent (respectively) to the L_2-inner product in the velocity space and to a discrete H^1-inner product in the pressure space, [81]. For this reason, we refer to this method as the $L^2 - H^1$ preconditioner.

The preconditioner is highly efficient for this type of mixed system. It is able to handle very heterogeneous permeability fields. In this work, the MINRES-$L^2 - H^1$ solver has been applied for fine grid problems. Unfortunately it does not perform well on systems upscaled with the version of AMGe used in this work. This is due to the spectral properties of the velocity mass matrix changing when it has been upscaled. We will demonstrate this in the numerical results in Section 3.3.

3.1.2 $H(\mathbf{div}) - L^2$ preconditioner

The mixed system

$$\begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix} = \begin{bmatrix} \mathbf{F_u} \\ F_p \end{bmatrix} \tag{3.3}$$

can be solved using an augmented Lagrangian method

$$\begin{bmatrix} M + B^T W^{-1} B & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix} = \begin{bmatrix} \mathbf{F_u} + B^T W^{-1} F_p \\ F_p \end{bmatrix}. \tag{3.4}$$

This approach can deal with ill-conditioned or even singular M matrices, [20]. We precondition the system in (3.4) with the following block diagonal preconditioner

$$\mathbf{Q}^{-1} \begin{bmatrix} M + B^T W^{-1} B & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix} = \mathbf{Q}^{-1} \begin{bmatrix} \mathbf{F_u} + B^T W^{-1} F_p \\ F_p \end{bmatrix}, \tag{3.5}$$

$$\mathbf{Q}^{-1} = \begin{bmatrix} (M + B^T W^{-1} B)^{-1} & 0 \\ 0 & W \end{bmatrix}, \tag{3.6}$$

where $(M+B^T W^{-1} B)^{-1}$ is approximated with the Auxiliary Space AMG solver for $H(\mathrm{div})$ problems (ADS), [110], from Hypre, [53]. W is the weighted pressure mass matrix, which for the lowest order is a diagonal matrix containing the element volumes. Contrary to the $L^2 - H^1$ preconditioner, the $H(\mathrm{div}) - L^2$ preconditioner can be used to solve upscaled problems. However, if the permeability field is too heterogeneous, the method struggles both for finest grid problems and upscaled problems. As a finest grid solver, we have typically preferred MINRES-$L^2 - H^1$, since it is quite fast.

The $H(\mathrm{div}) - L^2$ method is typically used as a preconditioner for MINRES.

3.2 Preconditioners based on AMGe

An alternative to the block diagonal preconditioners is to use the hierarchy of coarse spaces computed by AMGe as interpolation operators in a multigrid algorithm. In the following, two AMGe based preconditioners will be described.

- AMGe multigrid preconditioner with Vanka smoothing.
- The Multilevel Divergence Free preconditioner.

We also tested Restricted Additive Overlapping Schwarz smoothing, but further studies are required to evaluate its performance.

3.2.1 AMGe multigrid with Vanka smoother

In the early stages of the thesis work, experimentations were carried out, where cell-based Vanka smoothing was combined with AMGe coarse spaces to form a multigrid solver. Vanka smoothing is a type of Gauss-Seidel smoothing specific for mixed systems. For element e, it can be formulated as

$$\begin{array}{ll} \text{Form local residual:} & \mathbf{r}_e = f_e - (A\mathbf{x})_e, \\ \text{Compute local correction:} & \mathbf{c}_e = A^{-1}\mathbf{r}_e, \\ \text{Update global solution:} & \mathbf{x}_e = \mathbf{x}_e + \omega\mathbf{c}_e, \end{array}$$

where A_e is the restriction of the mixed system A to the element e, f_e is the restriction of the right hand side f on e and ω is a dampening parameter. This operation is performed for each cell one at a time. The smoother is very attractive from an implementation point of view as it is simply a matter of extracting the portion of the system matrix corresponding to an element, inverting local saddle point problems and updating values in a vector. Therefore it can be applied to arbitrarily coupled equations without deeper knowledge about the underlying problem. The difficulties arise in parallel implementations, where for unstructured grids it faces the same problems as Gauss-Seidel smoothers do. The problem is that Vanka and Gauss-Seidel smoothers rely on the solution in neighbour elements to be updated. This is difficult in a parallel setting on unstructured meshes. Either a sophisticated coloring scheme is required or a hybrid Jacobi/Gauss-Seidel strategy is needed (this may disturb convergence).

3.2.2 Multilevel Divergence Free preconditioner

The first preconditioner is a specialized indefinite AMGe preconditioner coined the Multilevel Divergence Free preconditioner: MLDivFree. It is developed at LLNL. MLDivFree uses a hierarchy of AMGe coarse spaces to form a preconditioner for symmetric indefinite saddle point problems of the following form.

$$\begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix} \tag{3.7}$$

MLDivFree can be summarized in the following three actions:

1. Find $\hat{u}$ such that the divergence constraint $B\hat{u} = q$ is satisfied.
2. Find $u = \hat{u} + C\sigma$ such that $||M(\hat{u} + C\sigma) - f||^2_{M^{-1}} \rightarrow \min$, where C is the discretization of the curl operator (obtained by AMGe as explained in

Chapter 2).

3. Find p such that $||B^T p - Mu - f||^2_{M^{-1}} \rightarrow \min$. This is the dual operation of step 1.

In practice, this is implemented by a symmetric V-cycle with a sophisticated multiplicative smoother. The pre-smoothing involves first solving for each agglomerate a local saddle point problem. Next a divergence free correction is obtained by solving for $\delta u = C(\delta\sigma)$, where $\sigma \in H(\text{curl})$ is computed by applying some smoothing iteration to the linear system $C^T M C\sigma = C^T f$ (Hiptmair smoothing). The post-smoother consists of the same two components but in the reverse order. The GMRES method is used for acceleration.

3.3 Numerical results

In the following a number of studies comparing the block diagonal preconditioners to the AMGe based preconditioners are presented. First, a study comparing the Multilevel Divergence Free preconditioner to the two block diagonal preconditioners $L^2 - H^1$ and $H(\text{div}) - L^2$ for a single linear solve of the Darcy model problem

$$\begin{cases} K^{-1}\mathbf{u} - \nabla p = 0 \\ \nabla \cdot \mathbf{u} = 0 \end{cases} \Rightarrow \begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.8}$$

is presented. This study is followed up by two similar studies, but where the linear systems are part of the solution scheme for a two-phase reservoir simulator.

3.3.1 AMGe vs. L^2-H^1 vs. $H(\text{div}) - L^2$

This section contains a comparison between the Multilevel Divergence Free preconditioner, the $L^2 - H^1$ preconditioner and the $H(\text{div}) - L^2$ preconditioner on the Darcy model problem. The preconditioners are tested both for a structured mesh and an unstructured mesh.

For the structured mesh, we use the SPE10 permeability field, [95] illustrated in Figure 3.1. The original finite element mesh consists of $60 \times 220 \times 85 = 1,122,000$ elements. We impose a pressure gradient along the x-direction and no flux conditions on the other faces.

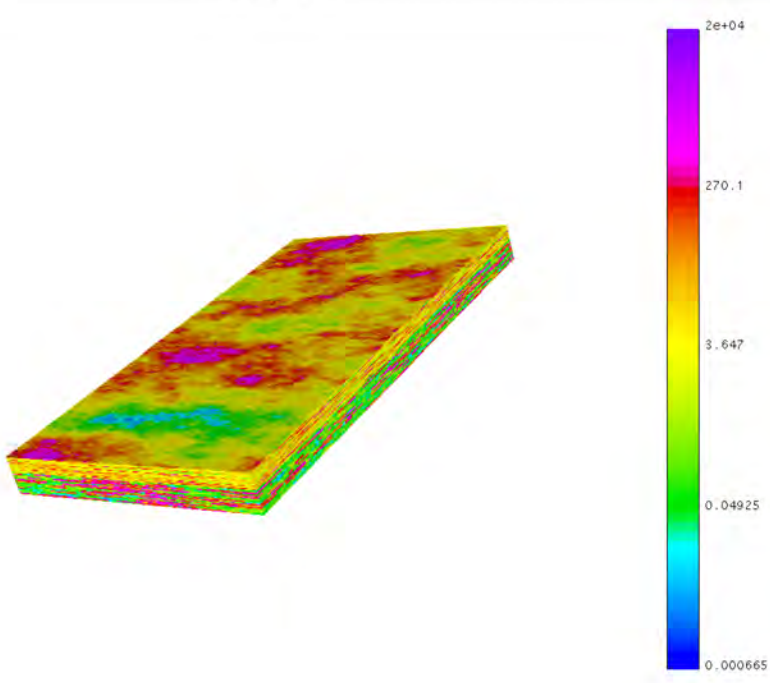
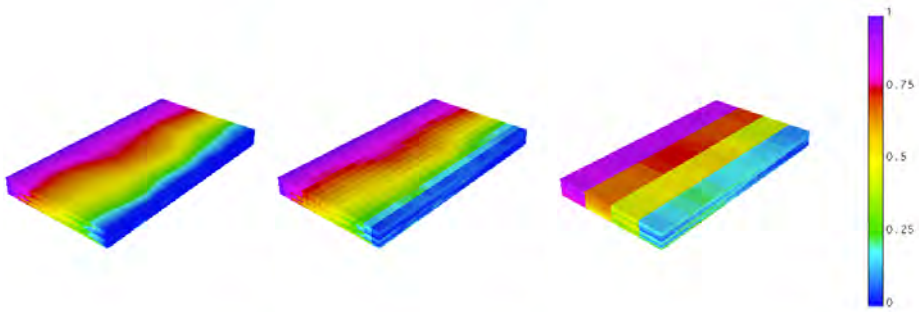


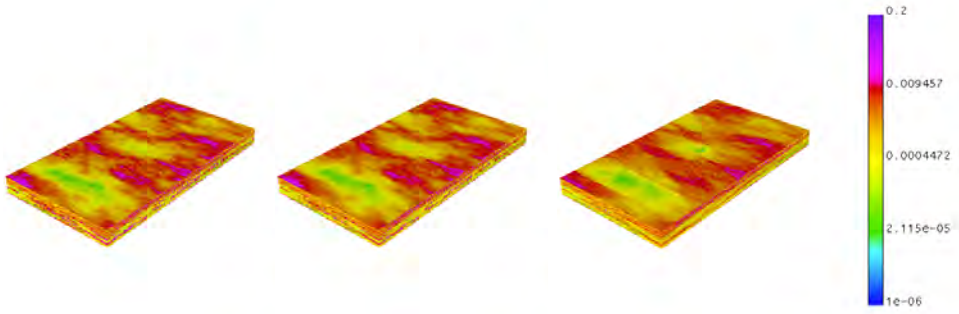
Figure 3.1: SPE10 permeability field in the x-direction

The mesh is agglomerated twice using a structured semi-coarsening. AMGe coarse spaces have been computed to construct the upscaled problems. The finest grid solution and the two upscaled solutions for the pressure and velocity are plotted in Figures 3.2 and 3.3.



(a) 1.1M DoF **(b)** 70K DoF **(c)** 4.7K DoF

Figure 3.2: (a): finest grid pressure solution. (b) and (c): AMGe upscaled pressure solutions.



(a) 3.4M DoF **(b)** 220K DoF **(c)** 16K DoF

Figure 3.3: (a): finest grid velocity solution. (b) and (c): AMGe upscaled velocity solutions.

The computational time as a function of degrees of freedom is plotted in Figure 3.4. The left plot shows the setup times and the right plot shows the solve time. All three solvers appear to be optimal for this structured grid problem (with structured agglomeration). The MINRES-$L^2 - H^1$ solver is clearly the fastest both in terms of setup and solve times. Table 3.1 holds the number of linear iterations used for each solver. For the MINRES-$L^2 - H^1$ solver, the number of linear iterations is higher for grid level 1 than for grid level 0 and for grid level 2. This is consistent with other experiments, where we have found that the MINRES-$L^2 - H^1$ solver struggles in particular for the large upscaled systems.

On the contrary, the GMRES-MLDivFree and MINRES-$H(\mathrm{div}) - L^2$ solvers keep the number of iterations low.

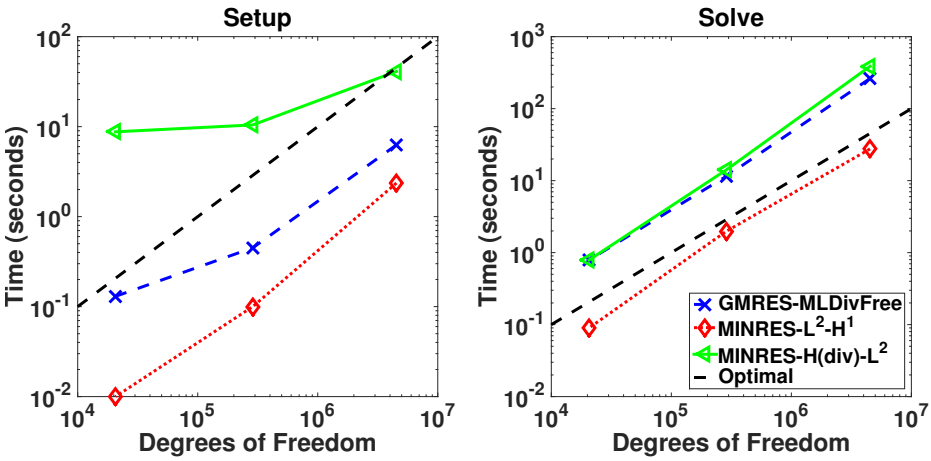
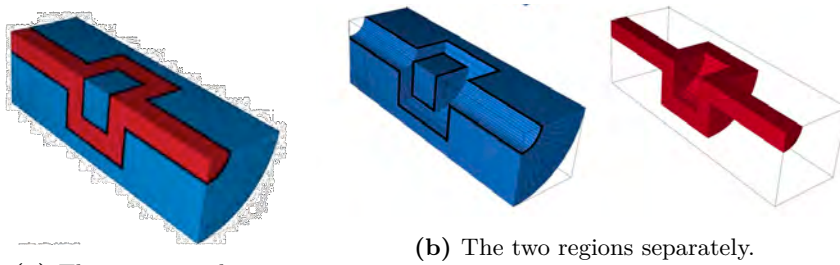


Figure 3.4: Computational time as a function of degrees of freedom.

Grid level	#DoFs	GMRES-MLDivFree its.	MINRES-$L^2 - H^1$ its.	MINRES-$H(\mathrm{div}) - L^2$ its.
Level 0 (fine)	4525000	27	42	47
Level 1	287275	18	60	28
Level 2	20626	17	47	26

Table 3.1: Degrees of freedom and the number of linear iterations for each solver.

This concludes the structured grid example. The next test case solves the same mixed model problem, however we now use the unstructured mesh in Figure 3.5 and we have a non-zero right hand side in parts of the domain (the red part). The mesh is separated into two regions. One regions has a high permeability value and the other region has a low permeability value. The difference between the two permeability values is $\frac{K_{\mathrm{blue}}}{K_{\mathrm{red}}} \approx 10^3$.

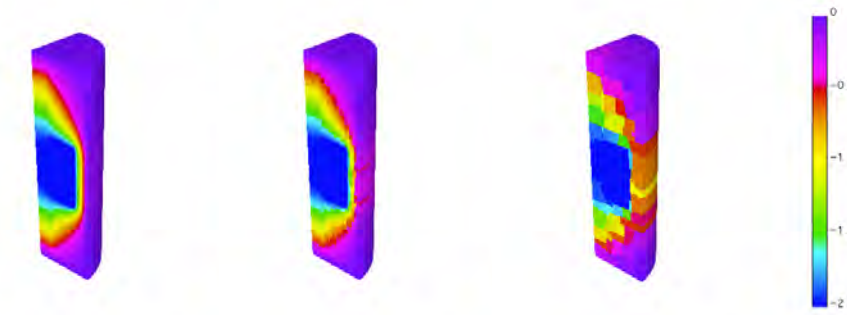


(a) The entire mesh.

(b) The two regions separately.

Figure 3.5: Unstructured mesh with a high and a low permeability region.

In the red region of the domain, the right hand side for the second equation is 1. In the blue regions of the domain the right hand side is still 0. We apply the boundary condition $\mathbf{u} \cdot \mathbf{n} = 0$ on the lateral surfaces and $p = 0$ on the top and bottom surfaces. The finest grid solution and the two upscaled solutions for the pressure and velocity are plotted in Figures 3.6 and 3.7.



(a) 117K DoF **(b)** 3.6K DoF **(c)** 121 DoF

Figure 3.6: (a): finest grid pressure solution. (b) and (c): AMGe upscaled pressure solutions.

(a) 362K DoF **(b)** 40K DoF **(c)** 1.8K DoF

Figure 3.7: (a): finest grid velocity solution. (b) and (c): AMGe upscaled velocity solutions.

The computational time as a function of degrees of freedom is plotted in Figure 3.8. Again, the left plot shows the setup times and the right plot shows the solve time. Table 3.2 holds the number of linear iterations used for each solver. As it is evident from the results, the $L^2 - H^1$ preconditioner performs very badly on the upscaled problems. For instance, the MINRES-$L^2 - H^1$ solver requires more than 2000 iterations to converge for the problem on grid level 1. This is due to the fact that the coarse velocity mass matrix in the upscaled problems is no longer uniformly spectrally equivalent to its diagonal. The solver based on the AMGe preconditioner: MLDivFree maintains optimal convergence behaviour. This is also the case for the MINRES-$H(\mathrm{div}) - L^2$ solver.

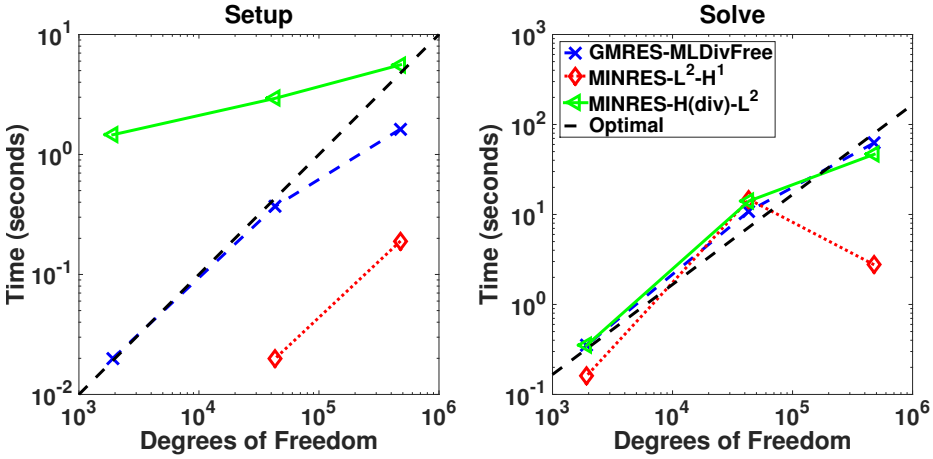


Figure 3.8: Computational time as a function of degrees of freedom.

Grid level	#DoFs	GMRES-MLDivFree its.	MINRES-$L^2 - H^1$ its.	MINRES-$H(\mathrm{div}) - L^2$ its.
Level 0 (fine)	478332	61	49	51
Level 1	43334	83	2068	73
Level 2	1916	70	564	60

Table 3.2: Degrees of freedom and the number of linear iterations for each solver.

3.3.2 Multilevel Divergence free preconditioner vs. L^2-H^1

Note that this study overlaps with Paper III, but it is included here for completeness. This study demonstrates how the $L^2 - H^1$ solver is efficient for finest grid problems, but do not perform well for the solution of the upscaled problems. However, the AMGe based preconditioner (Multilevel Divergence free) allows for robust and scalable preconditioning of both the finite element and upscaled linear systems. In this test, we consider the top 35 layers of the second dataset of the Tenth SPE Benchmark and we use the x-permeability as an isotropic permeability field. As part of a full reservoir simulator implementation, we are solving the mixed system in Paper III. The mixed system is almost identical to the mixed system introduced in the beginning of this chapter. The only

difference is that the coefficient $K^{-}1$ is instead $K^{-1}\lambda^{-1}$, where λ is the total mobility and accounts for relative permeabilities and viscosities. Also, wells are accounted for in these experiments. Computations are carried out in serial on a cluster with Intel Xeon EP X5660. The upscaled linear systems are constructed by the AMGe coarse spaces. An agglomeration factor of 4, 8, 16, 32, 64 and 128 fine elements per agglomerate is used in the computations to generate upscaled models at different levels of resolution.

To build the AMGe preconditioner, we recursively apply the upscaling procedure so that a nested hierarchy of agglomerated meshes and coarse spaces is constructed. In this experiment, we recursively apply the agglomeration algorithm with a coarsening factor of 16 "elements" (these are actually agglomerates themselves) per agglomerate until a reasonable small coarsest problem size is achieved (see Figure 3.9). In addition, to reduce the operator complexity (ratio of non-zeros on coarse levels compared to the finest level) of the AMGe preconditioner and obtain better numerical efficiency (at the cost of a slight increase in the number of iterations), we do not enforce approximation properties for the velocity space at the coarser levels of the AMGe hierarchy.

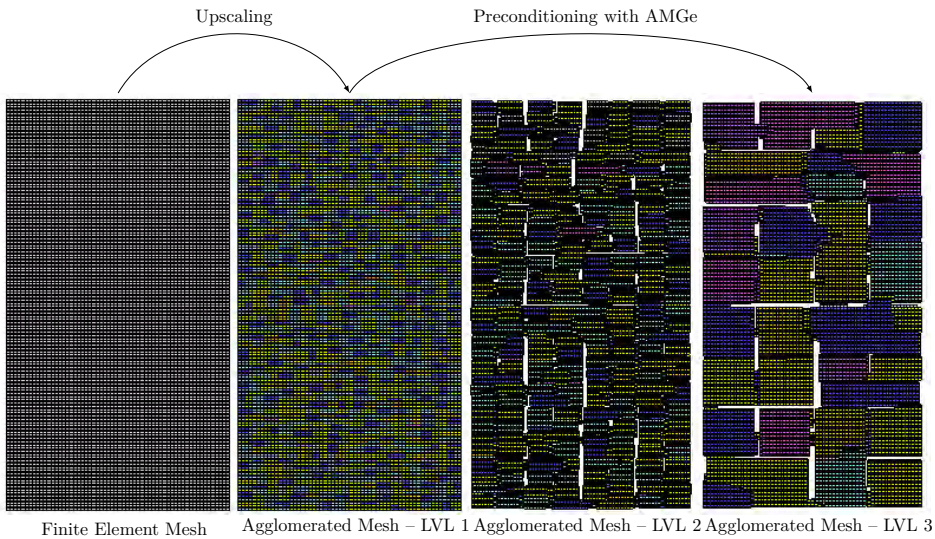


Figure 3.9: Full coarsening of the top 35 layers of the SPE10 benchmark case. The graph partitioner METIS is used. The agglomerated mesh at level 1 consists of roughly 32 fine grid elements per agglomerate and the agglomerated meshes at level 2 and 3 consist of roughly 16 finer grid agglomerates per agglomerate. A coloring algorithm is used to distinguish between agglomerates. Furthermore, agglomerates are slightly shrunk for visualization purposes.

Fig. 3.10 shows the computational time as a function of the number of degrees of freedom. Furthermore, Table 3.3 shows the information about the upscaled systems and the number of iterations to solve the problem with MINRES-$L_2 - H^1$ and GMRES-AMGe. For AMGe, the generation of coarse spaces, the solve itself and the setup cost are reported separately. We expect a significant reuse of the coarse spaces to be possible. The setup cost entails the factorization of small local matrices so that they can be reused at every GMRES iteration.

We observe that the performance of the $L_2 - H^1$ preconditioner significantly deteriorates for the upscaled problems. The reason of the drastic increase in the number of iterations (for the $L_2 - H^1$ preconditioner) and computational cost (the time to solution for the fine grid problem is almost the same as the one for the coarsest upscaled problem) is that the coarse velocity mass matrix in the upscaled problems is no longer uniformly spectrally equivalent to its diagonal. On the contrary, the AMGe preconditioner shows an optimal convergence behavior with respect to the number of unknowns, resulting in a significant reduction of the computational cost when solving the upscaled problem.

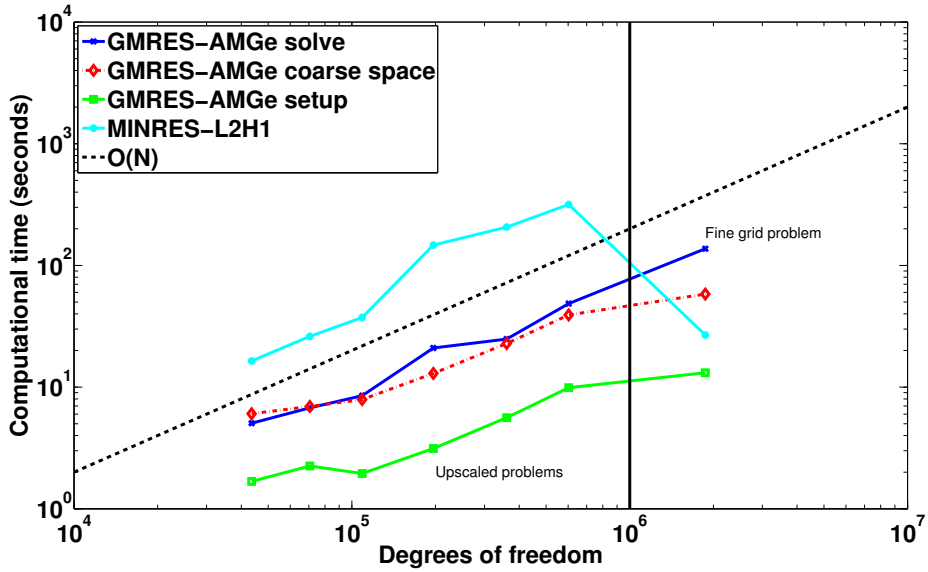


Figure 3.10: Computational time vs degrees of freedom for AMGe and $L_2 - H^1$ preconditioning of the upscaled systems.

	#elements	#faces	#DoFs	nnz	$L_2 - H^1$ iter.	AMGe iter.
Fine grid problem	462000	1409000	1871000	20813175	64	83
Fine elements per agglomerate						
4	104025	491319	602983	11150935	2593	53
8	57800	287056	360684	7297706	2999	44
16	28905	154599	196481	4370973	4167	66
32	14445	81542	108867	2719543	2039	44
64	7228	45008	70607	2403027	2294	50
128	3613	23626	43573	1885228	1447	31

Table 3.3: Information about the upscaled problem and the number of iterations needed to solve it.

3.3.3 AMGe with Vanka smoothing vs. $H(\mathbf{div}) - L^2$

This section holds a comparison between a two-level GMRES-AMGe with cell-based Vanka smoothing and MINRES-$H(\mathrm{div}) - L^2$ for the same mixed system as in the previous section. The only difference is that these tests are based on a gravity inversion setup as illustrated in Figure 3.11 and there are no wells. The top half of the domain is initialized with a water saturation of 1 and the bottom half of the domain is initialized with an oil saturation of 1 (red indicates high amounts of oil). Due to lower oil density than water density, the gravity will force the oil to switch place with the water. The permeability field is homogeneous with a permeability of 100 mD. Pressure is initialized to 0 bar.

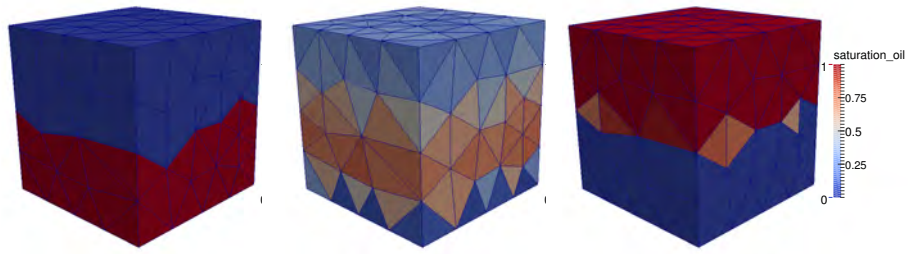


Figure 3.11: Illustration of gravity inversion setup.

Figure 3.12 shows the average number of linear iterations per nonlinear iteration for different problem sizes. GMRES-AMGe uses a similar number of iterations with growing problem size, whereas the number of iterations grow slightly for MINRES-$H(\mathrm{div}) - L^2$. Figure 3.13 shows the time-to-solution (for the full reservoir simulation) using a direct sparse solver, GMRES-AMGe and MINRES-$H(\mathrm{div}) - L^2$ for different problem sizes. As expected for GMRES-AMGe, the time-to-solution does not fully scale linearly with problem size, since only two

levels are employed meaning the size of the coarse systems grows with problem size and these are solved using a direct sparse solver. The time-to-solution for MINRES-$H(\mathrm{div}) - L^2$ scales (almost) linearly with problem size.

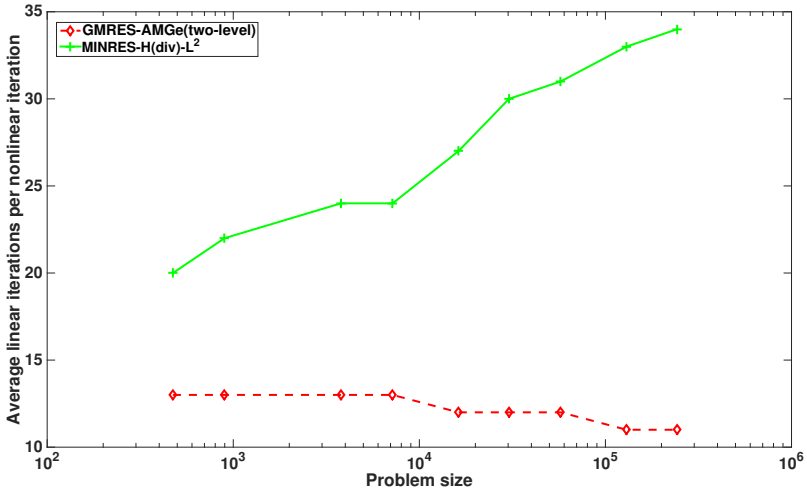


Figure 3.12: Comparison of average number of linear iterations per nonlinear iteration for GMRES-AMGe and MINRES-ADS. The domain size is fixed and fixed time step sizes are used. The grid consists of unstructured tetrahedrons

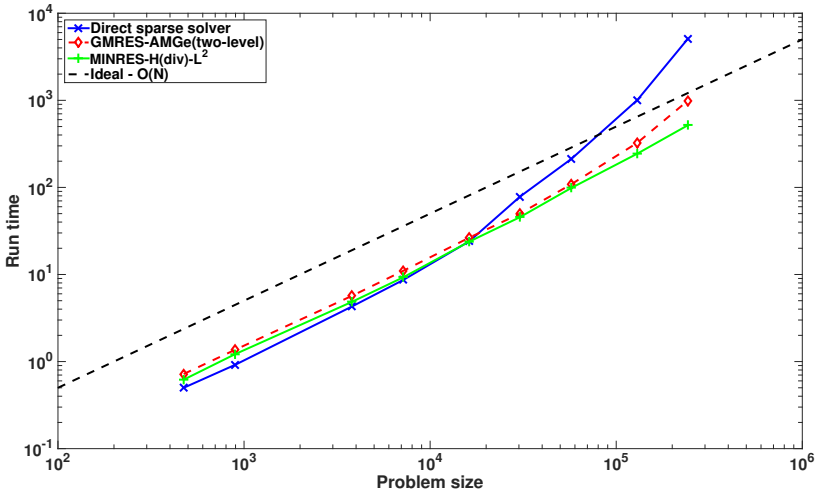


Figure 3.13: Comparison of time-to-solution using 3 different linear solvers: Direct sparse solver, GMRES-AMGe (two-level) and MINRES-ADS. The domain size is fixed and fixed time step sizes are used. The grid consists of unstructured tetrahedrons.

Chapter 4

Overview of papers

As part of the thesis work, four papers have been written. In this chapter, some introductory remarks to the papers will be given. Furthermore, the work carried out in the four papers will be linked together to shed some light on the reasoning behind each paper and how they tie together. The four papers are listed below.

- *Paper I - Nonlinear Multigrid for Reservoir Simulation.* The first paper describes the continuation of the work carried out in the authors M.Sc thesis (in collaboration with Klaus Langgren Eskildsen). In this work, an alternative solver scheme based on the Full Approximation Scheme (FAS) was proposed to try to improve on the typical Newton+GMRES+CPR(AMG+ILU) solver scheme. Both solver strategies are implemented along with a 3D three-phase compressible fluids/rock immiscible reservoir simulator. The two solver strategies are compared and some interesting results are found. Specifically, FAS provided
  - improved scalability with respect to problem size,
  - a larger basin of attraction allowing larger time steps to be solved for
  - and very fast reduction of residuals in the first few nonlinear iterations (FAS V-cycles). This is of great value in reservoir simulation, where only few digits of accuracy are needed and solving to very stringent tolerances is meaningless.

At the time, the thought was there that FAS could have a role in enabling an efficient implicit solver for so-called multiscale techniques. In Paper IV, initial steps towards this are demonstrated.

- *Paper II - Numerical Multilevel Upscaling for Incompressible Flow in Reservoir Simulation: An Element-Based Algebraic Multigrid (AMGe) Approach.* The second paper arises from a collaboration with Panayot Vassilevski and Umberto Villa at Lawrence Livermore National Laboratory. Together with Ilya Lashuk, they had been working on the implementation of a new version of AMGe with improved/guaranteed approximation properties. At that time, the author was searching for methods which would allow FAS to work on unstructured meshes. This was the case with AMGe. Before implementation of a fully implicit reservoir simulator based on FAS-AMGe, initial steps needed to be taken to understand the underlying methods. These initial steps resulted in this paper. Here an improved IMPES formulation is used to solve the two-phase incompressible fluids/rock and AMGe is used to introduce a multilevel variational upscaling tool for that particular model. Some of the key contributions specific to variational upscaling of the reservoir simulation equations introduced in the paper are

  - the ability to handle aggregates with non-planar faces, and achieve local mass conservation on all levels, and provide support for well models.
  - flexibility to assign a variable number of degrees of freedom per agglomerated face (interface between two agglomerated elements) that is automatically determined by the desired accuracy and by the topology of the agglomerated face by means of SVD.
  - leverages the components from a multigrid algorithm, using algebraically constructed coarse spaces and variational Galerkin coarsening. This allows reuse of coarse spaces for linear/nonlinear solver purposes.
  - ensuring that the coarse spaces maintain guaranteed order of approximation at all levels.

- *Paper III - Multilevel Techniques Lead to Accurate Numerical Upscaling and Scalable Robust Solvers for Reservoir Simulation.* The third paper builds on Paper II by demonstrating that the coarse spaces (interpolation operators between grid levels) can be used for both

  - variational upscaling, where the system of PDEs are numerically upscaled and therefore much faster to solve. This can be used for acceleration of uncertainty quantification and optimization. In a later chapter, it is demonstrated how Multilevel Monte Carlo based on

AMGe variational upscaling accelerates the convergence of Monte Carlo simulations.

- constructing a linear solver. In particular, it is demonstrated how a classic (and state-of-the-art) block diagonal preconditioner for mixed systems of this type becomes useless on upscaled system. This is due to the fact that the coarse velocity mass matrix in the upscaled problems no longer is uniformly spectrally equivalent to its diagonal. This is unfortunate since it effectively rules out a number of very good preconditioners for this type of mixed systems. To remedy this issue, we introduce in the paper a preconditioner based on the AMGe coarse spaces. The AMGe preconditioner shows an optimal convergence behaviour with respect to the number of unknowns regardless of whether the systems stem from a finest grid discretization or from upscaling.

Furthermore it is shown how one hierarchy of agglomerated meshes and associated coarse spaces can be built in a setup phase and parts of the hierarchy can then be used for variational upscaling purposes, while the coarser part of the hierarchy can be used to construct a solver for the upscaled problem. Imagine the hierarchy consisting of 5 grid levels. If a solution is needed for the second finest level (level 1), the coarse spaces from levels $2, 3, 4$ can then be used to construct a solver for level 1. Finally the paper demonstrates the strong scalability for a distributed parallel implementation of a finest grid solver scheme. Very good strong scaling is achieved for the SPE10 case.

- *Paper IV - Nonlinear Multigrid Solver Exploiting AMGe Coarse Spaces with Approximation Properties.* The final paper completes the circle by combining AMGe with FAS to construct a nonlinear solver for unstructured meshes that can be used to solve implicit variationally upscaled formulations. The previous attempts with the original AMGe versions without approximation properties were less successful due to less accurate interpolation. Using coarse spaces with approximation properties, the FAS approach on unstructured meshes should be as powerful/successful as FAS on geometrically refined meshes. The paper compares FAS-AMGe with Newton's method and Picard iterations. It is demonstrated that FAS is faster than Newton's method and Picard iterations for the experiments considered in the paper. The model in the paper is still simplistic (steady-state single phase to model primary depletion of an oil reservoir, where the permeability decreases exponentially with the pressure) and the next step would be to extend the model and apply FAS-AMGe to a black oil reservoir simulation formulation.

Paper I - Nonlinear Multigrid for Reservoir Simulation

Authors: Max la Cour Christensen, Klaus Langgren Eskildsen, Allan Peter Engsig-Karup and Mark Wakefield.

Published in SPE Journal (SPE 178428), 2015.

Chapter 5

Paper I - Nonlinear Multigrid for Reservoir Simulation

Abstract: A feasibility study is presented on the effectiveness of applying nonlinear multigrid methods for efficient reservoir simulation of subsurface flow in porous media. A conventional strategy based on global linearization via Newton's method is compared with an alternative strategy based on local linearization leading to a nonlinear multigrid method in the form of the full approximation scheme (FAS). It is demonstrated through numerical experiments that, without loss of robustness, the FAS method can outperform the conventional techniques in terms of algorithmic and numerical efficiency for a black-oil model. Furthermore, the use of the FAS method enables a significant reduction in memory usage when compared to conventional techniques, which suggests new possibilities for improved large-scale reservoir simulation and numerical efficiency. Lastly, nonlinear multilevel preconditioning in the form of a Hybrid-FAS-Newton strategy is demonstrated to increase robustness and efficiency.

5.1 Introduction

Reservoir simulation is an essential practice for evaluating and optimizing strategies to extract hydrocarbons and manage reserves. Based on recovery forecasts

from the simulator, different scenarios can be compared and a degree of risk and uncertainty analysis undertaken. In addition there is a continued requirement to be able to model at higher resolutions in order to fully utilize seismic data and improve the fidelity of the numerical results. For the reservoir simulator to perform this role effectively, it is crucial that it is robust, fast, and scalable. With modern hardware trends showing sustained progress around many-core architectures, the parallel programming paradigms and challenges associated with developing a simulator continue to evolve. As a part of this evolution it is relevant to reconsider whether conventional algorithmic strategies can be adapted to better utilize modern and emerging many-core architectures for massively parallel computations.

The numerical methods employed in conventional reservoir simulators are memory intensive and do not readily scale to either the latest large-scale distributed systems or modern and emerging many-core architectures. Modern hardware is designed with a limited amount of local memory on a per-core basis and often has a relatively high cost associated with data transfer. This cost is partially alleviated by high memory on-chip bandwidths but this is not the case for off-chip memory access. Consequently, applications that are limited by memory bandwidth constraints would benefit greatly from algorithmic redesign that leads to data-local implementations. The linear solver in a conventional reservoir simulator is known to be memory bandwidth constrained and motivated by this, data-local nonlinear multigrid methods that can exploit the high low-latency on-chip bandwidth on processors are investigated for both algorithmic and numerical efficiency.

Multigrid methods have successfully been employed in a wide range of research areas and have already proved to be effective in utilizing many-core hardware. There are several examples of distributed parallel implementations of multigrid. Algebraic multigrid methods have been parallelized to scale to a large number of cores [51, 32, 3], as have the geometric family of methods on structured grids [107]. Geometric multigrid methods on structured grids were the first to be implemented on massively parallel high-throughput graphical processing units (GPUs) [93, 79, 54]. More recently, efficient and scalable massively parallel implementations of geometric and algebraic multigrid methods on GPUs have been documented [4, 92]. Recent advances have also been made for geometric multigrid methods on unstructured grids [78, 96].

5.1.1 Conventional techniques

Fluid flow in reservoirs can be described mathematically with a system of partial differential equations (PDEs) governing subsurface porous media flow [16, 30].

With conventional techniques, it is common to use a global linearization Newton-type method to solve the strongly nonlinear system of equations arising from the spatial and temporal discretization of the governing equations [16]. This global linearization results in large linear systems, and hence the linear solver component can constitute more than 70% of the computational time of a simulation. Iterative linear solvers depend on effective preconditioners, which can be hard to parallelize to the extent required by many-core hardware [104]. Additionally, the memory requirement to store the sparse Jacobian for the linear systems is significant. This is not in line with modern hardware trends, which indicate local memory continues to be limited per core.

The linear solver methods frequently employed include preconditioned Krylov subspace methods such as ORTHOMIN, with nested factorization as the preconditioner [9]. A significant advance in preconditioning for the type of linear systems generated by a reservoir simulator was made with the Constrained Pressure Residual preconditioning (CPR) method [117, 55]. CPR preconditioning was developed specifically for reservoir simulation and acknowledges that the governing equations are of a mixed parabolic-hyperbolic type. By targeting the parabolic part of the system as a separate inner stage, the CPR method can achieve an improved convergence rate for the complete linear systems. Whilst use of the CPR method is largely restricted to reservoir simulation it is worth noting that conceptually the method is similar to the SIMPLE-type schemes designed for the Navier-Stokes equations in which the pressure and velocity components of the solution are targeted separately [100].

The effectiveness of CPR was demonstrated in [49], in which a linear solver based on CPR preconditioning using algebraic multigrid (AMG) to solve the first-stage pressure system outperforms a simulator based on ORTHOMIN preconditioned with nested factorization.

5.1.2 Nonlinear multigrid techniques

An alternative to the conventional techniques described previously is the family of nonlinear multigrid methods as outlined in [109] and [51], specifically the full approximation scheme (FAS), which is investigated in this paper. The FAS algorithm is a nonlinear method in which the linearization is performed locally, rather than globally. As a result, the FAS method can be implemented with a smaller memory footprint, enabling larger simulation models to be considered for a given computing resource. Moreover, the FAS algorithm is potentially better aligned with the emerging many-core architectures, which are expected to continue to have limited memory available per core and require algorithms that promote both data-locality and fine-grained parallelism. Provided the data

is local and there is sufficient parallelism the modern many-core architectures such as GPUs can obtain very high flop rates and memory throughput [103], which is beneficial for computationally intensive programs.

Whilst nonlinear multigrid methods have been constructed and modified for the Navier-Stokes equations [56], interestingly, very little work has been published on the use of the FAS method for reservoir simulation. In [86] convergence rates of two variations of nonlinear multigrid on a simple 2D immiscible two-phase homogeneous example without gravity are demonstrated. In this paper, it is demonstrated that nonlinear multigrid provides fast, grid-independent convergence behaviour and optimal complexity, implying the computational cost per time step per grid point is independent of the number of grid points.

This paper serves as a feasibility study of the FAS method when applied to the nonlinear PDEs governing subsurface flow in porous media. Firstly, a reservoir simulator based on conventional techniques employing global linearization, Newton's method, FGMRES as the linear solver and with CPR preconditioning has been implemented. Secondly, a multilevel FAS method has been incorporated into the simulator such that the Newton's iteration (with FGMRES+CPR solver) acts as the coarse grid solver. In this way, the simulator can work both with global linearization with Newton's method and in a multilevel fashion with local linearization in FAS. Furthermore, this allows for a Hybrid-FAS-Newton approach, where initially a few FAS iterations are taken to get close to the solution after which the simulator switches to global linearization with Newton's method to benefit from the the quadratic convergence offered by this method. This approach can be interpreted as nonlinear multilevel preconditioning.

The code is a single threaded serial application and solves the system of PDEs governing 3D three-phase flow of oil, water, and gas in porous media taking into account gravitational effects and wells with multiple perforations. The spatial discretization is via the classical finite volume method and temporal integration is via the backward Euler method.

5.1.3 Paper contribution

To our knowledge, very little work has been published on the use of the FAS method for reservoir simulation other than the work of [86]. A study on the feasibility of the FAS method for reservoir simulation is particularly relevant as many-core hardware is starting to be adopted by the industry and supported by commercial software packages. This paper is the first to present the FAS method for three-phase flow in 3D with gravity, wells and with a heterogeneous benchmark, extending the previous study of [86]. In this paper, numerical exper-

iments provide a fair comparison between the conventional and FAS multigrid techniques, as well as highlighting interesting properties of the FAS multigrid algorithm that show promise with respect to addressing some of the key challenges in reservoir simulation. Furthermore, it is demonstrated that by combining FAS with Newton's method in a hybrid strategy, the best of both methods can be obtained. To our knowledge, this paper is the first to present nonlinear multilevel preconditioning for these equations.

5.2 Governing equations

Following [108], the time-dependent mass conservation law for multiphase flow in a subsurface porous medium is given by

$$\frac{\partial(\phi(p)m_c)}{\partial t} + \nabla \cdot \mathbf{f}^\alpha(p, m_c) = \sigma_c, \tag{5.1}$$

where $\nabla\cdot$ is the divergence operator, m_c is the component molar density, p is pressure, σ_c is the source term for wells (if present), ϕ is porosity, $\mathbf{f}^\alpha = b^\alpha \mathbf{v}^\alpha$ is a vector flux function, where b^α is the phase molar density and $\mathbf{v}^\alpha$ is the Darcy phase velocity vector. The superscript α denotes the phase and the subscript c denotes the component.

For each phase, the velocities can be expressed in terms of Darcy's law in the form

$$\mathbf{v}^\alpha = -\mathbf{K}\frac{k_r^\alpha(p, m_c)}{\mu^\alpha(p)}(\nabla p - \rho^\alpha(p) g \nabla z), \tag{5.2}$$

where $\mathbf{K}$ is an absolute permeability tensor, $\mu^\alpha(p)$ is viscosity, $k_r^\alpha(p, m_c)$ is relative permeability, $\rho^\alpha(p)$ is density, and g the gravitational acceleration.

The system is closed by assuming volume balance through a constraint on the saturations

$$\sum_\alpha S^\alpha(p, m_c) = 1. \tag{5.3}$$

The properties used for all of the above-mentioned terms are described in any reservoir simulation book. No flow domain boundary conditions are used for all tests.

In this paper a black-oil model is considered with three components and three phases: oil, water and gas. Thus, the resulting coupled system (5.1)-(5.3) has four primary variables; namely the component molar densities m_o, m_w, m_g and the pressure p. This system is of mixed hyperbolic-parabolic type and therefore

is a natural target for a numerical strategy that deals with each characteristic individually.

The well implementation allows for multiple perforations. For the production wells a bottom hole pressure control is used and for the injection wells, a constant injection rate control is used. For both injection and production wells a nonlinear system is solved to find respectively, the bottom hole pressure and the flow rates into the perforations of the wells. For simplicity and robustness, an average wellbore fluid density is used in the production wells.

5.2.1 Simplifications

For the purposes of this paper, a number of simplifications have been made:

- The fluids are immiscible, such that oil cannot dissolve in the gas phase and gas cannot dissolve in the oil phase. In reservoir simulation, this scenario is referred to as a dead oil, dry gas system. However, the framework can be naturally extended to components that can be present in more than one phase. Moreover, phase phenomena such as gas dissolution in the reservoir are strongly effected by pressure which is generally well resolved by multigrid approaches due to its non-local nature.
- The effects of capillary pressure are not considered, however unlike some of the sequentially formulated multiscale techniques in which the solution of the pressure field is split from the transport of the saturations, the FAS method is fully implicit and hence the implicit saturations are available to compute the capillary pressure terms. In this respect capillary pressure can be treated as in a conventional fully implicit formulation.
- Porosities are kept constant throughout the domain. Permeability variations are expected to have a greater impact on the performance of the simulator than porosity, and for this reason the study is restricted to only considering the effects of permeability heterogeneity.

5.3 Discretization

A conservative discretisation of the model equations is done using a method of lines approach, where the finite volume method is used in space and the backward Euler method in time.

5.3.1 Spatial discretization

Let the domain of the considered oil reservoir be denoted $\Omega \in \mathbb{R}^3$ and discretized using N nonoverlapping grid cells or control volumes in a regularly structured grid. Let the ith grid cell be denoted Ω_i, $i \in \mathcal{C} = \{1, ..., N\}$, such that $\Omega = \bigcup_{i \in \mathcal{C}} \Omega_i$. The surface of Ω_i is denoted $\partial\Omega_i$. Since the grid is regularly structured, the volume V_i and the area A_i of Ω_i are denoted without the subscript in the following. The subscript $_{ij}$ indicates that properties are evaluated over the face between cell i and cell j.

By applying the finite volume method a semidiscrete system of differential equations can be derived from the differential form in (5.1)

$$\frac{\partial(V_p m_{c,i})}{\partial t} = s_{c,i} - f_i^\alpha, \quad i \in \mathcal{C}, \tag{5.4}$$

where $s_{c,i}$ is the source term of cell i and where f_i^α is the flux over the faces of a grid cell i in the direction of the outward normal and is given by

$$f_i^\alpha = \sum_{j \in \mathcal{N}^{(i)}} T_{ij} \lambda_{ij}^\alpha (\Delta p - \rho^\alpha g \Delta z)_{ij}, \quad i \in \mathcal{C}, \quad j \in \mathcal{N}^{(i)}, \tag{5.5}$$

where $\mathcal{N}^{(i)} \subset \mathcal{C}$ is the set of neighbour cells for cell i and the vertical spatial coordinate z increases downwards.

T_{ij} and λ_{ij}^α are respectively the transmissibility and the phase mobility as given below. Furthermore, the component mass conservation equation (5.4) has been multiplied by the pore volume $V_p = \phi V$, instead of explicitly considering porosity.

All the static and geometric quantities are combined into the transmissibility term

$$T_{ij} = \frac{A_{ij} k_{ij}}{\Delta h}, \quad i \in \mathcal{C}, \quad j \in \mathcal{N}^{(i)}, \tag{5.6}$$

where A_{ij} is the area of the face between cell i and cell j. The permeability on the face k_{ij} is approximated with the harmonic mean

$$k_{ij} = \frac{2 k_i k_j}{k_i + k_j}. \tag{5.7}$$

Similarly, all the properties dependent on either the pressure or the mass are combined into a single scalar called the mobility, denoted λ^α,

$$\lambda^\alpha = \frac{b^\alpha k_r^\alpha}{\mu^\alpha}. \tag{5.8}$$

Note, in the simplified black oil formulation considered in this paper where there is a single component in each phase the component mobility is the same as the phase mobility.

To ensure stability, the flux across a face is reconstructed using an upwind method defined as follows:

$$\lambda_{ij} = \begin{cases} \lambda_j & \text{if} \quad (\Delta p - \rho^\alpha g \Delta z)_{ij} < 0 \\ \lambda_i & \text{otherwise} \end{cases}, \quad i \in \mathcal{C}, \quad j \in \mathcal{N}^{(i)}. \tag{5.9}$$

Without upwinding, the numerical solution can display oscillations, overshoots, or undershoots; for example saturations less than zero or greater than one, or converge to an incorrect solution [16]. As a result of the upwinding, this is formally a first-order scheme.

5.3.2 Temporal integration

The implicit backward Euler method is used for temporal integration, enabling time steps of reasonable length to be realized. Applying this method to the semidiscrete system of equations in (5.4) leads to

$$V_p m_{c,i}^{n+1} = V_p m_{c,i}^n + \Delta t \left(s_{c,i}^{n+1} + f_i^{\alpha,n+1} \right). \tag{5.10}$$

This is the fully discretized system of mass conservation equations.

5.4 Conventional techniques in reservoir simulation

For the discretized equations presented in equation (5.10) and the saturation constraint in equation (5.3), the following residuals can be written:

$$\begin{aligned} r_{o,i}(m_o, p) &= \Delta t \left(s_{o,i} + f_i^o\right)^{n+1} + (V_p m_{o,i})^n - (V_p m_{o,i})^{n+1}, \\ r_{g,i}(m_g, p) &= \Delta t \left(s_{g,i} + f_i^g\right)^{n+1} + (V_p m_{g,i})^n - (V_p m_{g,i})^{n+1}, \\ r_{w,i}(m_w, p) &= \Delta t \left(s_{w,i} + f_i^w\right)^{n+1} + (V_p m_{w,i})^n - (V_p m_{w,i})^{n+1}, \\ r_{vb,i}(m_o, m_g, m_w, p) &= (S^o + S^g + S^w)^{n+1} - 1, \end{aligned}$$

where r_{vb} denotes the residual of the volume balance constraint.

This is a nonlinear system of equations, which can be solved using Newton's method. We formulate our nonlinear system of equations as

$$\mathbf{r}(m_o, m_g, m_w, p) = \mathbf{r}(\mathbf{x}) = \mathbf{0}, \tag{5.11}$$

and seek an improved approximation to the solution

$$\mathbf{r}(\mathbf{x} + \mathbf{h}) \approx \mathbf{r}(\mathbf{x}) + \frac{\partial \mathbf{r}}{\partial \mathbf{x}}\mathbf{h} = \mathbf{r} + \mathbf{J}\mathbf{h} = \mathbf{0}, \tag{5.12}$$

by solving iteratively for $\mathbf{h}$ in this linear system of equations $\mathbf{J}\mathbf{h} = -\mathbf{r}$, where $\mathbf{J}$ is the Jacobian for the nonlinear system of equations and $\mathbf{r}$ is the residual vector. We continue to iterate Newton's method until the residuals are satisfactorily small and it is in this loop that the vast majority of the computational time in a reservoir simulator is spent.

The conventional approach for linear solvers in reservoir simulation is to use a Krylov subspace method such as FGMRES preconditioned with the two-stage CPR method. CPR preconditioning involves a pressure predictor-corrector step for each linear iteration. The pressure system is formed by performing an IMPES–like reduction (implicit pressure explicit saturation). By treating saturations (or in our case molar densities m_c) explicitly, the fully coupled system consisting of $4x4$ block matrices can, via elementary row operations, be reduced to a scalar pressure system. For further details on this reduction procedure see [117, 55, 60].

The first stage of the CPR preconditioner is to solve for the pressure correction in Step 1 and one of the most popular methods in reservoir simulation is to use AMG. The second stage of the CPR preconditioner is often via ILU(0). ILU(0) deals effectively with high-frequency errors. However if ILU(0) is used as preconditioner by itself, the low-frequency error components linger and requires a large number of iterations to be sufficiently reduced [49]. However, if ILU(0) is used after the pressure correction, only the high-frequency errors remain since the first stage in CPR targets the low-frequency errors associated with the pressure system. Individually, the two components ILU(0) and AMG would be inefficient, but together they constitute a powerful preconditioner.

The CPR preconditioner was developed specifically for reservoir simulation and therefore it is not found in generic iterative linear solver software libraries. Consequently we have implemented the CPR preconditioner in the PETSc framework [17, 18, 19]. In the first stage of CPR we apply the AMG code BoomerAMG, [51], from the Hypre library, [53], and in the second stage we apply the ILU(0) implementation in PETSc.

5.5 The Full Approximation Scheme

In contrast to the global linearization methods, the FAS algorithm deals directly with the nonlinear system. The FAS algorithm exploits the fact that the two main components of multigrid, error smoothing and coarse grid correction, are also applicable to nonlinear problems [109].

5.5.1 The basics of multigrid methods

Multigrid methods are a class of defect-correction methods, based on calculating a correction term from a computed residual. This process is repeated iteratively until the solution is satisfactory. Geometric multigrid methods operate with multiple grid representations of the same domain. The concept is to define the problem on the finest grid and then interpolate (restrict) all the necessary information onto a coarser grid. This procedure is repeated recursively until the coarsest grid is reached upon which a residual equation is solved and a correction term obtained. The correction term is then recursively interpolated (prolongated) back to the finest grid, where the approximation to the solution is corrected.

An error smoothing operator is applied after each restriction and prolongation operation, to remove the high-frequency contributions to the error. Often, a couple of Jacobi-type or Gauss-Seidel-type iterations is sufficient. See [109] for more details on multigrid techniques.

5.5.2 Nonlinear multigrid method: FAS

Consider the nonlinear system

$$\mathbf{A}(\mathbf{u}) = \mathbf{f}, \tag{5.13}$$

where $\mathbf{A}(\mathbf{u})$ is a nonlinear operator that depends on $\mathbf{u}$. Here parentheses are used to indicate nonlinearity. The error is defined as $\mathbf{e} = \mathbf{u} - \mathbf{v}$ and the residual for the approximated $\mathbf{v}$ is defined as

$$\mathbf{r} = \mathbf{f} - \mathbf{A}(\mathbf{v}). \tag{5.14}$$

Inserting (5.13) in (5.14) gives the residual equation

$$\mathbf{A}(\mathbf{u}) - \mathbf{A}(\mathbf{v}) = \mathbf{r}. \tag{5.15}$$

As described in [51], the FAS method computes a coarse grid correction term based on the residual equation in (5.15), which using the error relation $\mathbf{u} = \mathbf{v}+\mathbf{e}$ can be rewritten as

$$\mathbf{A}(\mathbf{v}+\mathbf{e}) - \mathbf{A}(\mathbf{v}) = \mathbf{r}. \tag{5.16}$$

Assuming uniform grids, consider this equation on the coarsest grid with mesh size $H = 2h$; h being the mesh size of the finer grid one grid level up, (5.16) can be written as

$$\mathbf{A}_H(\mathbf{v}_H + \mathbf{e}_H) - \mathbf{A}_H(\mathbf{v}_H) = \mathbf{r}_H. \tag{5.17}$$

The coarse grid residual $\mathbf{r}_H$ is the restriction of the fine grid residual,

$$\mathbf{r}_H = I_h^H \mathbf{r}_h = I_h^H(\mathbf{f}_h - \mathbf{A}_h(\mathbf{v}_h)), \tag{5.18}$$

where I_h^H is the restriction operator. Similarly, the coarse grid approximation $\mathbf{v}_H$ is the restriction of the fine grid approximation $\mathbf{v}_h$. This is in contrast to linear multigrid where it is only necessary to restrict the residual. Using these definitions, (5.17) is written as

$$\mathbf{A}_H \underbrace{\left(I_h^H \mathbf{v}_h + \mathbf{e}_H\right)}_{\mathbf{u}_H} = \underbrace{\mathbf{A}_H\left(I_h^H \mathbf{v}_h\right) + I_h^H(\mathbf{f}_h - \mathbf{A}_h(\mathbf{v}_h))}_{\mathbf{f}_H}. \tag{5.19}$$

Since the right-hand side of (5.19) consists of known terms, the solution $\mathbf{u}_H$ to this equation can be determined. Based on this solution, the coarse grid correction term is computed as $\mathbf{e}_H = \mathbf{u}_H - I_h^H \mathbf{v}_h$. This correction term is prolongated one grid level up, where it is used to correct the solution to the residual equation at that grid level, and the process is repeated until the finest grid is reached. In the end an approximation to the nonlinear problem at the fine grid is found. Note that the linear solver is only applied at the coarsest grid and hence the full Jacobian matrix is only assembled for this grid level. This implies a large memory reduction compared to the conventional techniques.

5.5.3 Choice of multigrid components

The restriction stencil is an (x, y)-semicoarsening stencil that only restricts along the x- and y-direction. In reservoir simulation, the cells are typically thinner in the z-direction resulting in a stronger coupling along this axis. The restriction stencil averages the primary variables and sums the residuals of the four fine grid cells to obtain the value of the coarse grid cell. The prolongation stencil is simply an injection stencil, which transfers the value of one coarse grid cell into the four fine grid cells.

These are the simplest interpolation operators possible and for irregular grids more sophisticated methods exist in the literature. Currently no special considerations in the restriction and prolongation of cells containing wells have been

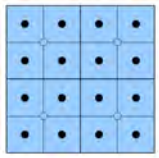


Figure 5.1: Cell-centered grid cells in fine 4×4 and coarse 2×2 grid, where ($\bullet$) indicate fine grid cell centers and ($\circ$) indicate coarse grid cell centers.

made, although it is thought that further improvements could be achieved by adjusting the weights of the interpolation functions in the vicinity of the wells, similar to the ideas introduced in [22].

The smoother used is a z-line Gauss-Seidel-Newton smoother. The smoother is constructed by locally assembling the Jacobian for the cells in the z-line, using the latest updated values of the primary variables available. Equation (5.20) shows the structure of the resulting block tridiagonal linear systems.

$$\overbrace{\begin{bmatrix} \mathbf{B}_1 & \mathbf{C}_1 & & & \mathbf{0} \\ \mathbf{A}_2 & \mathbf{B}_2 & \mathbf{C}_2 & & \\ & \mathbf{A}_3 & \mathbf{B}_3 & \ddots & \\ & & \ddots & \ddots & \mathbf{C}_{Nz-1} \\ \mathbf{0} & & & \mathbf{A}_{Nz} & \mathbf{B}_{Nz} \end{bmatrix}}^{\mathbf{J}_{line}} \overbrace{\begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \vdots \\ \mathbf{h}_{Nz-1} \\ \mathbf{h}_{Nz} \end{bmatrix}}^{\mathbf{h}_{line}} = - \overbrace{\begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \vdots \\ \mathbf{r}_{Nz-1} \\ \mathbf{r}_{Nz} \end{bmatrix}}^{\mathbf{r}_{line}}, \tag{5.20}$$

where the matrices $\mathbf{B}_1, ..., \mathbf{B}_{Nz}$ are 4×4 diagonal matrices containing partial derivatives with respect to their own cell and $\mathbf{A}_2, ..., \mathbf{A}_{Nz}$ and $\mathbf{C}_1, ..., \mathbf{C}_{Nz-1}$ are the 4×4 off-diagonal matrices containing partial derivatives with respect to neighbour cells (in the line).

Inside the line smoother, a single Newton iteration is performed, where the Jacobian is inverted efficiently using a block version of the Thomas algorithm. Every cell in the simulation belongs to one and only one z-line, and all the z-lines are processed in one smoothing operation. The smoother could be generalized for irregular grids by tracing lines through the model that seek to follow the higher transmissibility directions.

In our implementation, the nonlinear problem on the coarsest grid is solved using the same method as other simulators, namely Newton's method with FGMRES(30) preconditioned using the CPR method composed of AMG and ILU(0). Essentially, this means that FAS is wrapped around an existing solver

scheme: Newton-FGMRES-CPR(AMG-ILU(0)) and the only extra components needed are a smoother and the methods for interpolation between grids. This also allows for switching between Newton and FAS if needed.

All tests in this paper were performed with a single Newton iteration for the coarsest grid problem.

5.5.4 Hybrid FAS-Newton (nonlinear multilevel preconditioning)

As it will be demonstrated in the numerical results section, a hybrid between local linearization with FAS and global linearization with Newton's method has the potential to both increase robustness and efficiency of the nonlinear solver scheme. By experiments, it is found that FAS is better at reducing quickly the residual in the first few iterations. Furthermore, it is found that FAS is able to converge when the initial solution is further away from the true solution, compared to Newton's method. However, multigrid methods have asymptotically first order convergence, whereas Newton's method has second order convergence close to the solution. By taking 1 or 2 FAS V-cycles and then switching to Newton's method, it is possible to obtain the best of both methods, namely a larger radius of convergence and second order convergence. See Figure 5.2 for an illustration of the V-cycle followed by a couple of newton iterations.

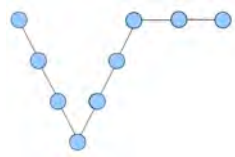


Figure 5.2: Illustration of Hybrid FAS-Newton. A V-cycle followed by two newton iterations.

Using this strategy is at the expense of the memory savings obtained with FAS, since now the Jacobian needs to be assembled on the fine grid mesh. However, if memory is not an issue on the hardware at hand, a hybrid approach may prove to be the strongest. Also by combining local and global linearization, any doubts as to whether FAS can handle local effects in the transport equations can be put at ease.

5.6 Experimental setting

A suite of different problems is considered:

- Gravity segregation in which the gas and water phases exchange place as a result of gravity segregation.
- Quarter five-spot model.
- Modified Egg Model.
- SPE10 top 35 layers[1].

For these models, the porosities were chosen to be constant throughout the domain in order to focus on how permeability contrasts affect the robustness and the performance of the simulators. Reservoir simulators tend to be more sensitive to heterogeneous permeability fields rather than heterogeneous porosity fields since the permeability forms part of the flow terms which couple the pressure and component variables. A heuristic time step size controller is used unless otherwise stated. The time step size is based on the number of Newton iterations or FAS cycles k used in the previous time step(s). Consequently, the time step size is a direct result of the convergence rate of the nonlinear solver. Specifically, we use the simple heuristic written in Algorithm 1. Lastly, the initial guess for both Newton and FAS is the solution at the previous time step.

Algorithm 2 Heuristic time step size controller

if $k < \frac{1}{2}k_{\max}$ **then**
    Success = Success + 1
    $\Delta t = \Delta t \cdot (1.0 + 0.1 \cdot \text{Success})$
else if $k \geq \frac{2}{3}k_{\max}$ **then**
    Success = 0
    $\Delta t = 0.5 \cdot \Delta t$
else
    Success = 0
end if

[1] http://www.spe.org/web/csp/

5.7 Numerical experiments

In this section the different nonlinear solvers are compared. The conventional techniques are referred to as the standard Newton (SN) or simply Newton's method in the following sections, and the FAS (and Hybrid FAS-Newton) method is referred to as FAS(ℓ) and (Hybrid-FAS(ℓ)), where ℓ indicates the number of grid levels employed.

For all the simulations, the stopping criteria and tolerances for the nonlinear solver are identical for both the Newton and FAS algorithms. An absolute stopping criteria was used with typical tolerances of 10^{-4} for mass conservation equations and 10^{-6} for the constraint on saturations (unless otherwise stated). The linear solver is by experimentation found to work well with an absolute stopping criteria of 10^{-12} and a relative stopping criteria of 10^{-10}. The timings are carried out on hardware with specifications given in Appendix 5.10.

5.7.1 Gravity segregation

5.7.1.1 Scalability

The initial conditions for these problems are to set the middle third of the cells to have an oil saturation of one, the bottom third to have a gas saturation of one, and the top third to have a water saturation of one. The pressure is initialized to 250 bar in all grid cells.

A heterogeneous permeability field was generated using trigonometric functions and a random number generator. The algorithm implemented generates a fixed permeability pattern within a user-specified range, regardless of the size of the problem. This approach enables simulations to be compared across a range of problem sizes. The generated permeability fields have higher variation in the z-direction than in the x- and y-directions, mimicking the layering of rock types often observed in reservoirs. The fixed pattern can be controlled by setting a seed for the random number generator. With the exception of the heterogeneity stress tests discussed in section 5.7.1.3, the permeability fields are generated with values ranging from 200 to 600 mD. In section 5.7.1.3 results are provided for more numerically challenging ranges of permeability values.

Figure 5.3 displays the total simulation run time as a function of the number of grid cells for a fixed reservoir size. Fixed time step sizes of one day are used to give a fair comparison of the performance of the nonlinear solvers, namely the

SN solver and the FAS solver.

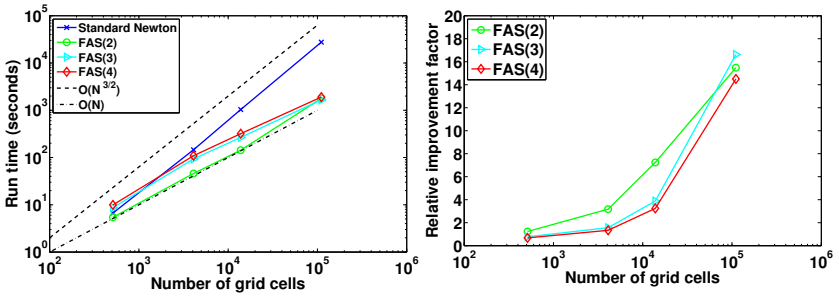


Figure 5.3: Left plot: Total run time as a function of the number of grid cells for SN, FAS(2), FAS(3), and FAS(4). The reservoir size is fixed at 480m $\times$ 240m $\times$ 48m. Fixed time stepping with $\Delta t = 1$ day. Simulation period is 100 days. Right plot: Relative improvement for FAS(ℓ) over SN.

The FAS-based reservoir simulator outperforms the SN-based reservoir simulator in terms of run time, with the margin growing as the problem sizes increase due to improved scaling of work. By maintaining a fixed coarse grid problem size, the FAS method is found to have linear scaling of work effort with increasing fine grid problem sizes. The SN based method scales asymptotically as $O(N^{3/2})$ due to the ILU(0) component. Additionally, for the SN method the number of Newton iterations per time step grows with increasing problem sizes. Also, we observe for the SN solver that the number of linear iterations per Newton iteration increases with increasing problem size too.

To analyze the convergence behaviour of the SN method compared to the FAS method, a number of experiments are carried out. For different numbers of grid cells and a fixed reservoir size, simulations are performed with fixed time step lengths of $\Delta t = 0.5$ days and $\Delta t = 1$ day for 100 days, and the average number of outer iterations per time step is computed. The outer iterations are taken to be Newton iterations for the SN method and FAS-cycles for the FAS method. Table 5.1 shows the results.

The results in Table 5.1 indicate that the convergence rate of the FAS method does not deteriorate as the resolution increases. This is not the case for the SN method, where the average number of Newton iterations increases as the resolution increases. This result is interesting because it addresses a limiting issue in conventional reservoir simulators in which the convergence rate of Newton's method deteriorates as the number of grid cells increases for a fixed reservoir size. Similar behaviour is observed for other applications [35]. It is expected

	Problem size			
$\Delta t = 0.5$	$8 \times 8 \times 8$	$16 \times 16 \times 16$	$24 \times 24 \times 24$	$48 \times 48 \times 48$
SN	3.2	3.3	4.3	8.1
FAS(2)	1.9	1.7	1.2	1.4
FAS(3)	2.9	3.2	2.3	1.6
FAS(4)	3.5	3.4	2.5	1.8
$\Delta t = 1.0$				
SN	4.0	6.1	9.4	20.1
FAS(2)	2.2	2.2	1.8	2.5
FAS(3)	3.6	4.2	3.3	2.5
FAS(4)	4.2	4.6	3.9	2.9

Table 5.1: Test of algorithmic effiency. Average number of outer iterations (Newton iterations for SN and FAS cycles for FAS) for fixed time step sizes $\Delta t = 0.5$ days and $\Delta t = 1.0$ days.

that the convergence rate of the FAS method is not bounded by the convergence rate of Newton's method [24]. This is also evident from the results in Table 5.1.

Note that reservoir simulators often use a more sophisticated Newton solver with dampening such as modified Appleyard [119]. Similar dampening techniques are equally applicable to the FAS method and in order to facilitate a fair comparison, a standard Appleyard chop for safeguarding has been implemented in both the FAS and SN methods.

In practice, a significant implication of the results above is that by using FAS it is possible to take larger time step sizes for finer resolution problems. This is a step towards the ability to choose time step sizes based on accuracy requirements only. Table 5.2 shows the number of time steps used when employing a heuristic time step size controller for a simulation period of 150 days with a maximum of 10 Newton iterations or FAS cycles allowed per time step and a fixed reservoir size.

	Problem size			
Method	$8 \times 8 \times 8$	$16 \times 16 \times 16$	$24 \times 24 \times 24$	$48 \times 48 \times 48$
SN	68	101	152	201
FAS(2)	25	33	48	57
FAS(3)	65	79	70	58
FAS(4)	314	97	88	69

Table 5.2: Number of time steps for 150 days of simulation using a heuristic time step size controller based on the number of Newton iterations or FAS-cycles for previous time steps. A maximum of 10 Newton iterations or FAS-cycles are allowed per time step.

The results in Table 5.2 demonstrate that by selecting an appropriate number of grid levels in FAS it is possible to maintain a similar number of time steps when the number of grid cells increases.

5.7.1.2 Memory comparison

As previously discussed, the memory requirement of the FAS algorithm is significantly less than that for conventional methods based on a global linearization. Figure 5.4 shows actual memory measurements of the two simulators implemented in this work.

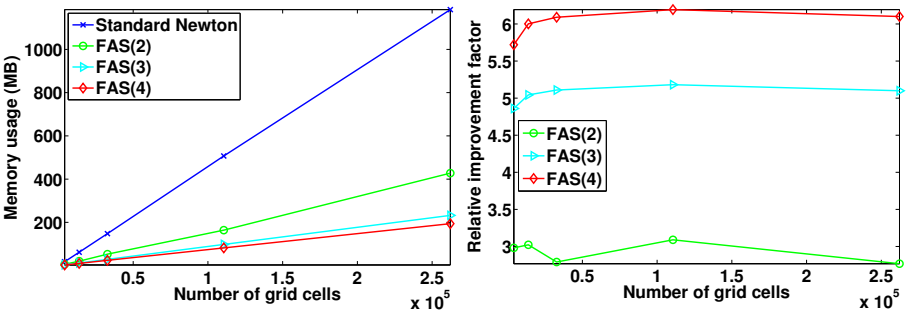


Figure 5.4: Left plot: Memory usage as a function of the number of grid cells. Right plot: Relative improvement for FAS(ℓ) over SN.

Figure 5.4 shows that the memory usage scales linearly with the number of grid cells for both simulators, as expected. However, compared to the SN based simulator, the FAS-based simulator uses 3-6 times less memory depending on the number of grid levels employed. For very large problems and a higher number of grid levels, the memory savings would be even more significant. In scenarios where memory usage is the limiting factor in model size, a FAS-based algorithm would therefore raise the operating envelope by a similar factor. In addition reducing the total memory usage leads to improved utilisation of memory bandwidth of hardware and as a result improves numerical efficiency for bandwidth-limited applications.

5.7.1.3 Heterogeneity stress test

A significant challenge in reservoir simulation is the efficient solution of models with highly heterogeneous permeability fields. As noted the performance

of a linear solver can be strongly affected by permeability heterogeneity and anisotropy, and addressing this issue ultimately led to the development of the CPR method [49]. The CPR approach handles the effects of the heterogeneity arising in the linear system. However the nonlinear solver is also affected by the permeability heterogeneity, often requiring smaller time steps to ensure that the nonlinear system converges.

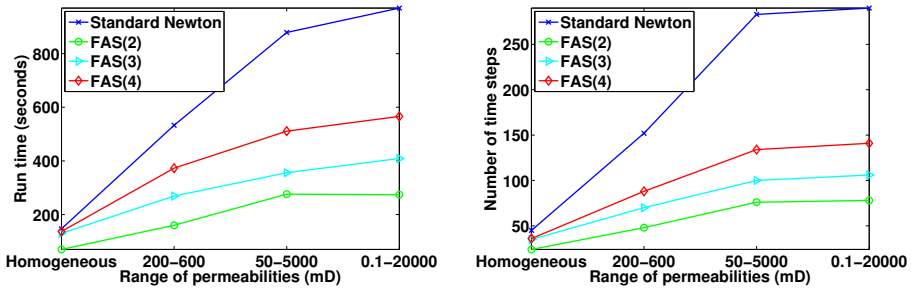


Figure 5.5: Left plot: Run time as a function of range of permeabilities. Right plot: Number of time steps as a function of range of permeabilities. Problem size: $24 \times 24 \times 24$, 5 pre- and post-smoothings, 150 days, cell sizes: $20m \times 10m \times 2m$.

Figure 5.5 demonstrates results from a "stress test", in which the range of permeabilities is varied between two extremes. The two extremes are a homogeneous permeability field and a permeability field with values ranging from 0.1 to 20,000 mD. The left plot shows the run time for the SN method and the FAS method with 2, 3, and 4 grid levels. The corresponding number of time steps are displayed in the right plot.

The results obtained with the FAS algorithm results are encouraging, exhibiting superior run time and algorithmic performance. For these simulations the heuristic time step size controller is used.

5.7.2 Quarter five-spot pattern

The quarter five-spot pattern test case is chosen for its symmetry. It will be used to demonstrate scalability of FAS vs. Hybrid-FAS vs. Newton's method. In all cells, we initialize water saturation to 0.2 and oil saturation to 0.8. Pressure is initialized to 250 bar. The producer is controlled by bottom hole pressure and the injector is controlled by rate.

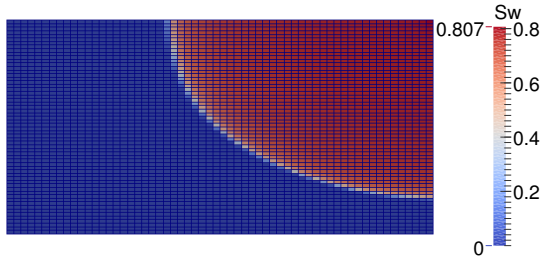


Figure 5.6: Quarter five-spot model. Water is injected in the top right corner and a producer is located in the bottom left corner.

Figure 5.7 compares the oil and water production between the FAS based simulator and a commercial reservoir simulator[2]. The commercial reservoir simulator uses 378 time steps with the initial time step length set to 0.1 days and with a maximum of 30 days time step lengths. Default solver settings are used. The FAS based simulator uses 384 time steps. For FAS, we use a relative tolerance of 10^{-6} and an absolute tolerance of 10^{-8}.

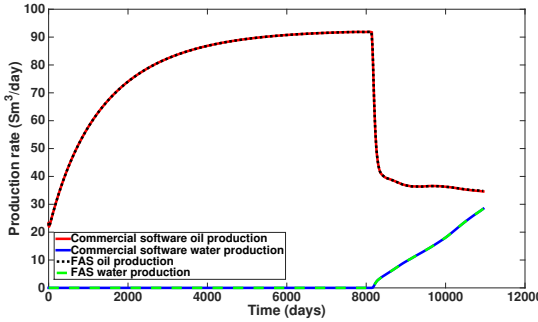


Figure 5.7: Quarter five-spot model validation against a commercial reservoir simulator. Grid is $24 \times 24 \times 10$ with a domain size of 1000m x 500m x 20m. Porosity = 0.3. Permeability = 100mD. Bottom hole pressure for producer: 200 bar. Injection rate: 100 Sm^3/day.

Figure 5.8 displays the results of a scaling test for the quarter five-spot model. The domain size is fixed and we increase the number of cells. With FAS we can

[2]ECLIPSE 100

take larger time step sizes compared to Newton's method, however the work per time step is also much higher in this case.

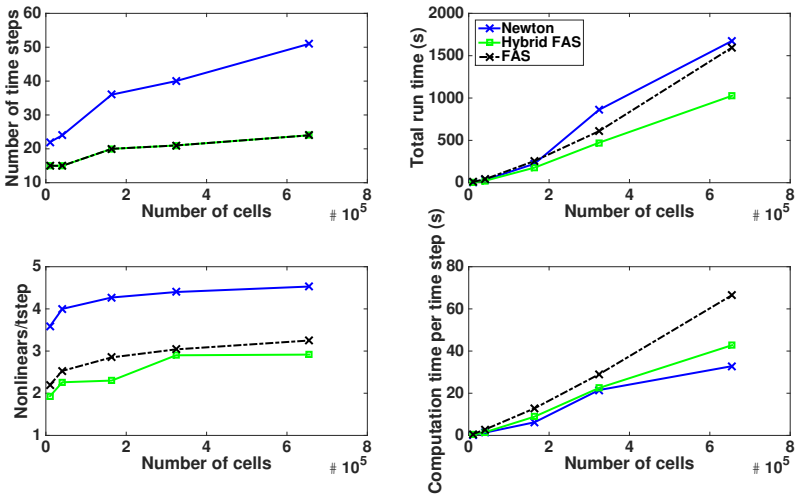


Figure 5.8: Quarter five-spot model scaling test.

By using a Hybrid-FAS-Newton strategy, we can decrease the amount of work per time step and still maintain the same low amount of time steps as seen in FAS.

5.7.3 Modified Egg model without inactive cells

A modified Egg model is included in the examples to demonstrate the capability of FAS to handle modest difficulty permeability fields and a larger number of wells with ease. The Egg model is a synthetic reservoir model consisting of an ensemble of 101 relatively small three-dimensional realizations of a channelized reservoir produced under water flooding conditions with eight water injectors and four producers, [57]. For this purpose we are only using the default realization of the permeability field that is given in the dataset.

Figure 5.9 displays the permeability field in the x- and y-directions. The z-direction has the same permeability field, only it is multiplied by a factor of 0.1. The modifications made to the Egg model are the following:

- The inactive cells are made active. We have not yet implemented this

feature in our simulator although its addition is not anticipated to be problematic since the prolongation and injection operators could be restricted to just considering the active cells within their stencils.

- The reservoir is initiated to a pressure of 400 bar everywhere and a water saturation of 0.1.
- Well indices are set to 1 for all wells. We have not yet implemented a formula for the well connection factor.
- The fluid properties are entered using a tabulated input rather than based on a Taylor expansion around the reference pressure.

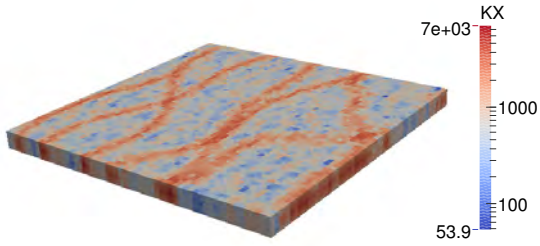


Figure 5.9: Egg model permeability field in the x- and y-directions.

Figure 5.10 shows the rates for each of the production wells simulated using the commercial reservoir simulator and FAS. With an initial time step size of 0.1 days and a maximum time step size of 30 days, the commercial reservoir simulator took 191 time steps and the FAS based simulator took 142 time steps.

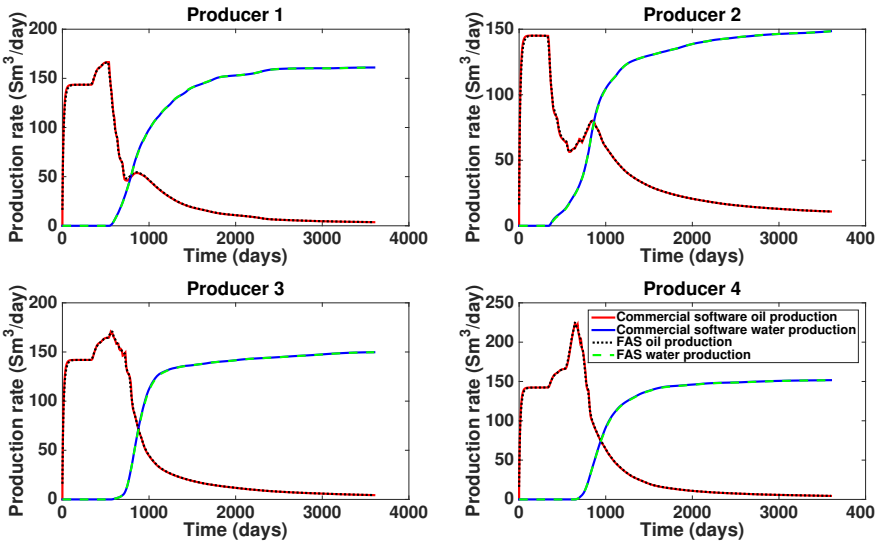


Figure 5.10: FAS validated against a commercial reservoir simulator for the (modified) Egg model.

Table 5.3 shows a comparison between FAS, Hybrid-FAS and Newton for simulation of the Egg model. In this experiment, we removed the restriction on a maximum time step size to see which solver is able to time step the longest.

Method	Time steps	Run time (s)	Time/tstep (s)	Nonlinears/tstep
Hybrid-FAS(3)	79	148	1.87	3.42
Hybrid-FAS(2)	82	145	1.77	2.95
FAS(2)	84	253	3.01	3.76
Newton	111	170	1.53	4.63

Table 5.3: Performance study between FAS, Hybrid-FAS and Newton's method for simulation of the Egg model.

As it is evident from Table 5.3, both FAS and Hybrid-FAS are able to employ longer time step sizes compared to Newton's method. However, FAS is for this case too computationally heavy per time step compared to Newton's method. Hybrid-FAS is clearly the fastest strategy. It should be noted that most of the failed time steps in these simulations for FAS and Hybrid-FAS were caused by failure to converge the nonlinear well solve for the injectors. If a more robust well solve is implemented, we believe FAS and Hybrid-FAS will be able to time step even longer. It should also be mentioned that FAS(3) needed particularly small time steps in order to converge for this model. We observed that the

coarse grid correction given by FAS(3) with this setup of wells gives a good residual reduction in the first few iterations, but then later reduces the residual slowly. This warrants further study and a possible solution could be to take into account the wells in the restriction and prolongation.

5.7.4 Top 35 layers of the SPE10 permeability field

A test case with the top 35 layers of the SPE10 permeability is included, 6.2, field to test FAS in the case of extreme permeability variations. The FAS solution was again validated with that of a commercial reservoir simulator.

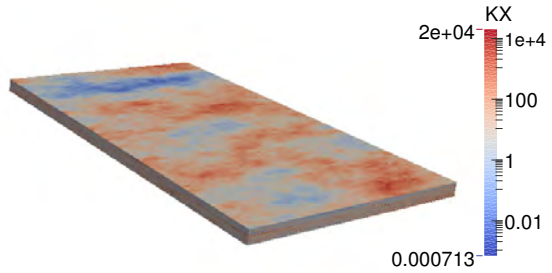


Figure 5.11: Top 35 layers of SPE10 x-permeability field.

The model was simulated for 100 days with an initial time step size of 0.1 days and a comparison between Newton's method, FAS and Hybrid-FAS was made. The solvers were allowed to time step as far as the convergence rate allowed for.

Method	Time steps	Run time (s)	Time/tstep (s)	Nonlinears/tstep
Hybrid-FAS(2)	15	392	26.13	2.6
Hybrid-FAS(3)	16	555	34.69	3.59
FAS(2)	15	932	62.13	3.8
Newton	44	1119	25.43	4.43

Table 5.4: Performance study between FAS, Hybrid-FAS and Newton's method for a simulation using the top 35 layers of the SPE10 permeability field.

Table 5.4 shows that Hybrid FAS again is able to time step the longest and still maintain a small time per time step.

5.8 Conclusion

The work presented in this paper focuses on an investigation of the applicability of nonlinear multigrid techniques, specifically the FAS method, for the efficient and scalable simulation of subsurface multiphase flow in porous media. A FAS solver has been wrapped around Newton's method in a reservoir simulation code. This enabled fair benchmarking between three different nonlinear solver strategies, namely global linearization with Newton's method, local linearization with FAS and a hybrid local/global strategy with FAS and Newton's method. State-of-the-art methods are used for the linear solver component, namely FGMRES with CPR preconditioning based on AMG and ILU(0).

It has been demonstrated that, for the given model equations and range of problems considered, the FAS/Hybrid-FAS-Newton method outperforms Newton's method in terms of algorithmic efficiency, computation time, memory requirements and robustness. These tests have been conducted with both homogeneous and highly heterogeneous permeability fields. Furthermore, it has been demonstrated that FAS is able to deal efficiently with wells with multiple perforations. We expect further improvements of FAS are possible by introducing restriction and prolongation operators which take into consideration the wells.

In section 5.7.1.1, it is demonstrated that the convergence rate of the FAS method is grid independent with increasing problem size. This in itself is an interesting observation since the Newton-type methods often suffer from reduced time step lengths when simulating at high resolution.

The reservoir simulator based on the FAS method has been developed as a serial program as a proof of concept. A parallel implementation and improved version of the FAS method on a distributed system is the subject of on-going work and will provide a basis for many further investigations. In addition, removing the simplifications in the models such as the lack of capillary pressure and a relatively simple fluid model need to be explored to ensure the results presented extend to more complex scenarios. Furthermore, extending this work to support unstructured meshes is the subject of on-going work.

5.9 Input values

Except for the Egg model, Tables 5.5 and 5.6 contain the physical constants and reservoir specific data used to generate the results presented in this paper. To obtain the same units for the accumulation and flow terms, a so-called Darcy con-

stant is multiplied with the transmissibilities and hence with the flow term. The Darcy constant is a unit conversion factor with a value of $C_{\text{Darcy}} = 0.00852702$ $^{\text{cP.m}^2}/_{\text{day/bar}}$.

Symbol	Value	Units
ρ_{sc}^{o}	800.0	$^{\text{kg}}/_{\text{m}^3}$
ρ_{sc}^{g}	0.9907	$^{\text{kg}}/_{\text{m}^3}$
ρ_{sc}^{w}	1022.0	$^{\text{kg}}/_{\text{m}^3}$
M^{o}	120.0	$^{\text{kg}}/_{\text{kmol}}$
M^{g}	25.0	$^{\text{kg}}/_{\text{kmol}}$
M^{w}	18.025	$^{\text{kg}}/_{\text{kmol}}$
g	0.0000980665	$^{\text{m}^2\text{kg}}/_{\text{bar}}$

Table 5.5: Physical constants.

Symbol	Value	Units
φ	0.3	-
p_{ref}	250	bar
B_{pref}^{w}	1.03	-
μ_{pref}^{w}	0.3	cP
C_v	0.0	$^{1}/_{\text{bar}}$
C	0.000041	$^{1}/_{\text{bar}}$
C_{rock}	0.000053	$^{1}/_{\text{bar}}$

Table 5.6: Input values.

5.10 Hardware specifications

Run time measurements were carried out on the hardware specified in table 5.7.

CPU:	Intel Xeon X5550 (quad-core) 2.66 GHz
Cache:	8 MB L3
FSB speed:	1333 MHz
Instruction set:	64-bit
Hard drive:	500 GB SATA (7200 RPM)
Memory:	24 GB
OS:	Scientific Linux 6.1

Table 5.7: Hardware specifications

Paper II - Numerical Multilevel Upscaling for Incompressible Flow in Reservoir Simulation: An Element-Based Algebraic Multigrid (AMGe) Approach

---

Authors: Max la Cour Christensen, Umberto Villa, Allan Peter Engsig-Karup and Panayot Vassilevski.

Submitted to SIAM Journal of Scientific Computing, 2014.

Chapter 6

Paper II - Numerical Multilevel Upscaling for Incompressible Flow in Reservoir Simulation: An Element-Based Algebraic Multigrid (AMGe) Approach

Abstract: We study the application of a finite element numerical upscaling technique to the incompressible two-phase porous media total velocity formulation. Specifically, an element-agglomeration based Algebraic Multigrid (AMGe) technique with improved approximation properties [73] is used, for the first time, to generate upscaled and accurate coarse systems for the reservoir simulation equations. The upscaling technique is applied to both the mixed system for velocity and pressure and to the hyperbolic transport equations providing fully upscaled systems. By introducing additional degrees of freedom associated with non-planar interfaces between agglomerates, the coarse velocity space has guaranteed approximation properties. The employed AMGe technique provides

coarse spaces with desirable local mass conservation and stability properties analogous to the original pair of Raviart-Thomas and piecewise discontinuous polynomial spaces, resulting in strong mass conservation for the upscaled systems. Due to the guaranteed approximation properties and the generic nature of the AMGe method, recursive multilevel upscaling is automatically obtained. Furthermore, this technique works for both structured and unstructured meshes. Multiscale Mixed Finite Elements exhibit accuracy for general unstructured meshes but do not in general lead to nested hierarchy of spaces. Multiscale multilevel mimetic finite differences allow for nested spaces but are usually less accurate in the flux construction for unstructured grids: our AMGe approach can be seen as a rigorous bridge that merges the best properties of the latter two. The accuracy and stability of the studied multilevel AMGe upscaling technique is demonstrated on two challenging test cases.

6.1 Introduction

Upscaling of geological properties is an essential practice in reservoir simulation, since the spatial resolution of the geological model often is too high for reservoir simulators to execute in practical times. The traditional approach employed today resorts to computing effective properties of the subsurface (permeability/porosity) by homogenization (averaging) techniques. Homogenization is formally an averaging of processes (and or mathematical operators), and hence, it goes far beyond averaging of parameters. This is an important and generally understated point that can help understanding how AMGe, and other multilevel operator-based approaches, are moving towards this direction. Many techniques are available for homogenization. The so-called flow-based upscaling methods are among the most used ones. They are typically based on solving a simple steady-state elliptic differential equation. Given the solution of this equation, effective coarse permeabilities can be computed. Some flow-based upscaling methods provide full tensor coarse permeabilities, but in practice mostly diagonal tensor permeabilities are used. These effective coarse properties are then perceived as the "true" model from this point on. However, the use of homogenization in the workflow introduces a black box step, where the relation/difference between the solution of the upscaled model and the solution of the fine grid model (geological resolution) is difficult (or impossible) to determine. A significant body of research has gone into improving this workflow by introducing new upscaling methods, which take into account more information of the underlying problem. In the petroleum engineering community, these methods are typically referred to as multiscale methods.

The methods described in this paper are strongly related to the Multiscale

Mixed Finite Element Method (MsMFEM). MsMFEM stems from early work described in [52] and [31], where specific finite-element basis functions were used to construct a tool for multiscale solution of elliptic partial differential equations in both primal and mixed form. Since then, much research has been carried out on this topic to improve the approximation properties and extend the range of physical phenomena described by the models, [10, 12, 11, 5, 70]. Among other things, the method was extended to achieve locally mass conservative velocity fields on the subgrid scale, which enabled a combination of MsMFEM and streamline simulations, [7]. Other work focuses on updating the multiscale basis functions for time dependent problems [69] or to capture specific features of the flow [6]. Multiscale methods have also attracted attention for locally conservative mimetic finite difference methods, [8], and for finite volume methods, [59, 39]. Adaptive strategies for multiscale techniques have also been proposed,[39, 75].

Multiscale methods have been extended for mimetic finite differences to work in a multilevel way for two-phase flow problems, [74, 75, 76]. Multiscale multilevel mimetic methods, namely M^3, have several similarities to our approach, and a few differences. Important similarities include the ability to handle aggregates with non-planar faces, and achieve local mass conservation on all levels, and provide support for well models. The AMGe approach possesses all these properties with the additional flexibility to assign a variable number of degrees of freedom per agglomerated face (interface between two agglomerated elements) that is automatically determined by the desired accuracy and by the topology of the agglomerated face by means of SVD.

Finally, the multilevel upscaling technique introduced in [80] leverages the components from a multigrid algorithm, using algebraically constructed coarse spaces and variational Galerkin coarsening. Our AMGe approach exhibits similar features with the additional caveat that we ensure that the coarse spaces maintain guaranteed order of approximation at all levels.

In parallel, to the above cited multiscale methods, in the algebraic multigrid (AMG) community, the construction of coarse problems was an essential component to develop efficient multilevel solvers for the fine-grid (fine-scale) problems of interest, especially in the unstructured mesh setting. It was recognized for quite some time, that an efficient two-grid (TG) solver requires as a necessary condition a coarse space that admits certain weak approximation properties. For a rigorous proof of this fact we refer to [43]. This fact can be viewed as a cornerstone motivating point in using AMG-constructed finite element coarse spaces as discretization spaces, i.e., as a tool for numerical upscaling. We refer the interested reader to the overview in [115] for more details. For some early work on using operator-dependent (AMG) coarse spaces for numerical homogenization, we refer to [68]. Among many AMG-type coarse spaces that can

be constructed, we are interested in ones that can handle general classes of finite element spaces, and hence be applicable to broad classes of PDEs; namely, the spaces that form a de Rham complex (i.e. the sequence of H^1-conforming, $H(\text{curl})$-conforming, $H(\text{div})$-conforming and L^2-conforming spaces) with applications to elliptic PDEs, Maxwell equations, Darcy flow equations, etc. The work [99], although specifically motivated to construct coarse de Rham complexes for use in multigrid solvers, provided the basis for extensions finalized in [72], to build coarse spaces with guaranteed approximation properties, giving rise to an efficient upscaling tool. Since our construction of coarse spaces applies to the entire de Rham sequence, the developed technique can also be used for other applications such as the mixed formulation of the Brinkman problem, [111, 116].

Often multiscale methods for reservoir simulation solve for the pressure (and velocity) on a coarse scale and keep the saturation equations on the fine grid. With this approach, solving the saturation equations quickly becomes the dominant bottleneck. Methods have been developed to also upscale the saturation equation, [120, 41]. Our approach enables upscaling of not only the mixed system for velocity and pressure, but also the transport equations for the saturations using the same framework. In the present paper, the coarse space used for the pressure is reused for the saturation equations. For problems involving quantities of interest that do not require a fine-scale solution of the saturation equations, such fully upscaled models can greatly accelerate standard methods in uncertainty quantification (e.g. using Multilevel Monte Carlo [37, 67]) and optimization (e.g. MG/OPT [21]) due to the ability to simulate with good accuracy at different levels of resolution with a reduction in computational cost equivalent to the reduction in the degrees of freedom for the upscaled models. AMGe has been developed (in LLNL) since its introduction ([61], [112], [27, 28], [71]) as a general multilevel coarsening framework with a wide range of applications. In addition to numerical upscaling, it is also designed to create efficient and optimal solvers that can be adapted throughout the simulation, [64]. Furthermore, the AMGe upscaling technique targets general unstructured meshes as well as higher-order elements. As such it is distinctly different (as being more general) than the above referred multiscale and mimetic methods.

For large-scale models and for applications such as uncertainty quantification and optimization, two-grid upscaling is insufficient. If the fine grid problem contains a large number of elements and more aggressive coarsening is needed to keep the upscaled problem small, constructing coarse spaces becomes increasingly expensive due to the growing sizes of the local flow problems. Furthermore, in uncertainty quantification (e.g. multilevel Monte-Carlo methods) and optimization, it is beneficial to employ a hierarchy of coarser systems to accelerate the convergence. For this reason, multilevel upscaling is a natural choice, since it enables the size of the local flow problems, needed to construct the coarse spaces,

to remain small (even fixed) and it provides a hierarchy of coarser discretizations. The AMGe technique employed in the present work allows a completely recursive upscaling, where a hierarchy of agglomerated meshes is used and due to the fact that we upscale the full system of equations, simulations can be carried out for a range of coarser representations, where the computational cost of a simulation is reduced in a similar fashion as the number of degrees of freedom and the non-zeros are reduced. The multilevel upscaling makes it possible to reuse the hierarchy of coarser spaces for different purposes. It can be used for both numerical upscaling, linear solvers and even nonlinear solvers, [26].

The contribution of the present paper is in the application of one version of AMGe with guaranteed approximation properties, [73, 72, 99], to the incompressible two-phase flow equations for reservoir simulation. Moreover, the framework is completely recursive, allowing for multilevel upscaling with guaranteed approximation properties. The paper demonstrates multilevel upscaling for two challenging test cases. In addition to upscaling the mixed system for pressure and velocity, the saturation equations are also upscaled with the same coarse spaces used for the pressure. To our knowledge there is no other method that supports mixed finite element formulations on general unstructured grids; it allows for multilevel nested hierarchies as well as it allows for great flexibility in the construction of the coarse spaces - with two possible strategies to locally enrich the coarse spaces by either using finer agglomerates or adding additional degrees of freedom for each agglomerate.

The remainder of the paper is structured as follows. In Section 6.2, we introduce the governing equations and present the weak formulation of the problem. In Section 6.3 the system of equations are discretized in space and time. In Section 6.4, the improved Element-based Algebraic Multigrid is introduced. Finally, in Section 8.6 numerical results are presented for two challenging test cases. The paper concludes with a summary and perspectives in Sections 6.6 and 6.7.

6.2 Governing equations

In this section, we briefly introduce a total velocity formulation of a simplified model for multiphase flow in porous media. We refer to [30] for a rigorous derivation of this formulation from the physics principles of mass and momentum conservation. The unknowns of the formulation are the total velocity $\mathbf{u}$, the pressure p, and the set S of the saturations S_α for each phase α. For example, we can have $\alpha = o, w, g$, hence $S = \{S_o, S_w, S_g\}$, where o stands for oil, w stands for water, and g stands for gas. Neglecting the effects of capillary pressure and

assuming incompressible rock and fluids, the system of equations is given by

$$\mathbf{K}^{-1}\lambda^{-1}(S)\mathbf{u} + \nabla p = \left(\sum_{\alpha} \rho_\alpha f_\alpha(S)\right) g\nabla z \tag{6.1}$$

$$\nabla \cdot \mathbf{u} = q(S,p) \tag{6.2}$$

$$\phi\frac{\partial S_\alpha}{\partial t} + \nabla \cdot \mathbf{u}_\alpha(S,\mathbf{u}) = \frac{q_\alpha(S,p)}{\rho_\alpha}, \tag{6.3}$$

where $\nabla\cdot$ is the divergence operator, $\mathbf{K}$ is the absolute permeability tensor, ρ^α is the mass density, ϕ is the porosity, g is the gravitational acceleration, and ∇z stands for the coordinate vector in z-direction. In this work, the capillary pressure is ignored for simplicity. Multiscale methods taking into consideration capillary pressure have been developed [98]. Since the terms and variables introduced in a formulation with capillary pressure are covered by the spaces that form a de Rham complex, we expect AMGe to be able to handle this as well.

The total velocity $\mathbf{u}$ is the sum of the phase velocities $\mathbf{u}_\alpha$

$$\mathbf{u} = \sum_{\alpha} \mathbf{u}_\alpha. \tag{6.4}$$

Similarly, the total source term $q(S,p)$ is the sum of the phase source terms q_α

$$q = \sum_{\alpha} \frac{q_\alpha(S,p)}{\rho_\alpha}. \tag{6.5}$$

The total mobility λ is given by

$$\lambda = \sum_{\alpha} \lambda_\alpha = \sum_{\alpha} \frac{k_{r,\alpha}(S_\alpha)}{\mu_\alpha}. \tag{6.6}$$

Here, for simplicity, we assume the "straight relative permeabilities" model and we let the relative permeability $k_{r,\alpha}$ be equal to S_α. The fractional flow function f_α is given by

$$f_\alpha = \frac{\lambda_\alpha(S_\alpha)}{\lambda(S)}. \tag{6.7}$$

Finally, the phase velocity $\mathbf{u}_\alpha$is related to the total velocity $\mathbf{u}$ by

$$\mathbf{u}_\alpha = (\mathbf{u} + \mathbf{G}_\alpha)\, f_\alpha, \tag{6.8}$$

where

$$\mathbf{G}_\alpha = \sum_{\substack{\beta=o,w,g \\ \beta\neq\alpha}} \lambda_\beta(\rho_\alpha - \rho_\beta)\mathbf{K}g\nabla z \tag{6.9}$$

allows for counter-current flow due to gravity.

The total velocity formulation can be advantageous as it allows for a less coupled system compared to other formulations [30, p. 25]. Unlike the individual phase mobility, the total mobility is positive definite, meaning its inverse always exists. Furthermore, the total velocity is smoother than a phase velocity. For these reasons, it is a good choice as a primary variable.

6.2.1 Wells

To model injection and production wells we use the well-understood Peaceman equations [101], which impose a linear relationship between the bottom hole pressure p_{bh} and the injection/production rates q_α. These formulas hold for vertical wells in anisotropic rock media with multiphase fluids and gravity and read

$$q_\alpha = \mathcal{WI}\lambda_\alpha(p_{\text{bh}} - p - \rho_\alpha g(z_{\text{bh}} - z)), \tag{6.10}$$

where z_{bh} is the well datum level depth, and z is the depth. The well index $\mathcal{WI}$ is a lumped parameter that takes into account the geometry of the perforation, the geometry of the cell, and anisotropy of the permeability tensor.
In our simulator, we use the bottom hole pressure, p_{bh}, to control the production wells, whereas we prescribe a rate for injection wells. In practice, this means

$$q_\alpha = \begin{cases} \textit{rate} & \text{if injector} \\ \mathcal{WI}\lambda_\alpha(p_{\text{bh}} - p - \rho_\alpha g(z_{\text{bh}} - z)) & \text{if producer,} \end{cases} \tag{6.11}$$

where *rate* is a user-given input. For producer wells, this means we have a dependence on pressure in the reservoir, which should be accounted for in the numerical schemes.

For completeness, we also provide the well-known formula to compute the well index $\mathcal{WI}$ derived in [101] under some simplifying assumptions on the structure of the permeability tensor and on the cell geometry. For a more thorough description of well models for Finite Element Methods, see [30, p. 450], while for more complex well models we refer to [40].
Assuming that the permeability tensor has a diagonal structure, $\mathbf{K} = \text{diag}(k_x, k_y, k_z)$ and that the element containing the well is a cuboid fully perforated in the vertical direction, the Peaceman formula for the well index reads

$$\mathcal{WI} = \frac{2\pi h_3 \sqrt{k_x k_y}}{\ln(r_e/r_w) + s_k}. \tag{6.12}$$

Here, s_k is the skin factor used to model formation damage from drilling, r_w is the well radius, and the equivalent radius r_e is

$$r_e = \frac{0.14\left(h_1^2\sqrt{k_y/k_x} + h_2^2\sqrt{k_x/k_y}\right)}{0.5\left((k_y/k_x)^{1/4} + (k_x/k_y)^{1/4}\right)}, \tag{6.13}$$

h_1, h_2 and h_3 being the mesh sizes in the x-, y- and z-directions.

6.2.2 Weak formulation

We now introduce some notation used throughout the paper. Let Ω be a bounded connected domain in $\mathbb{R}^d$ with a regular (Lipschitz continuous) boundary $\partial\Omega$, which has a well-defined unit outward normal vector $\mathbf{n} \in \mathbb{R}^d$. For the cases considered in this paper, $d = 3$.
For the vectorial functions $\mathbf{u}, \mathbf{v} \in \mathbf{L}^2(\Omega) = [L^2(\Omega)]^d$ and scalar functions $p, w \in L^2(\Omega)$, we define the inner products $(\mathbf{u}, \mathbf{v}) = \int_\Omega \mathbf{u}\cdot\mathbf{v}\, d\Omega$ and $(p, w) = \int_\Omega p\, w\, d\Omega$. Finally, we introduce the functional space $H(\mathrm{div};\Omega)$ defined as

$$H(\mathrm{div};\Omega) := \{\mathbf{u} \in \mathbf{L}^2(\Omega) \mid \mathrm{div}\,\mathbf{u} \in L^2(\Omega)\}.$$

For simplicity, we assume a no-flow boundary condition for the total velocity $\mathbf{u}$ and we impose

$$\mathbf{u}\cdot\mathbf{n} = 0. \tag{6.14}$$

This boundary condition is the most widely used in reservoir simulation, however different boundary conditions can be easily accommodated, including Dirichlet boundary condition for the pressure, or more sophisticated conditions obtained by coupling the equation of reservoir with an aquifer model.

We finally introduce the functional spaces $\mathcal{R}$ and $\mathcal{W}$, which are defined as

$$\mathcal{R} \equiv \{\mathbf{u} \in H(\mathrm{div};\Omega) \mid \mathbf{u}\cdot\mathbf{n} = 0 \text{ on } \partial\Omega\};$$
$$\mathcal{W} \equiv L^2(\Omega).$$

To derive the weak formulation for the mixed system in equations (9.5) and (9.6) we multiply equations (9.5) and (9.6) with the test functions $\mathbf{v} \in \mathcal{R}$ and $w \in \mathcal{W}$ and integrate over the domain Ω. After integration-by-parts of the non-conforming terms, applying the boundary condition $\mathbf{u}\cdot\mathbf{n} = \mathbf{v}\cdot\mathbf{n} = 0$, we obtain the following variational problem

PROBLEM 5 *Find* $(\mathbf{u}, p) \in \mathcal{R} \times \mathcal{W}$ *such that*

$$\begin{cases} \left(\mathbf{K}^{-1}\lambda^{-1}(S)\mathbf{u}, \mathbf{v}\right) - \left(p, \nabla \cdot \mathbf{v}\right) = \left(g\nabla z \sum_{\alpha} f_\alpha(S)\rho_\alpha, \mathbf{v}\right), & \forall\ \mathbf{v} \in \mathcal{R} \\ \left(\nabla \cdot \mathbf{u}, w\right) - \left(q(p,S), w\right) = 0, & \forall\ w \in \mathcal{W} \end{cases}$$

In Problem 11, the total mobility λ and the fractional flow function f_α depend on the phase saturations S, while, for the producer wells, q has a dependence on pressure p and the saturations S as given by equation (6.11). Problem 11 is well posed: the pair $(\mathcal{R}, \mathcal{W})$ is *inf-sup compatible* and $\frac{\partial q}{\partial p} \leq 0$ for all pressure p and saturations S.

In a simpler way, the weak formulation for the conservation law is derived by multiplying equation (9.7) with the test function $w \in \mathcal{W}$ and integrating over the domain Ω.

PROBLEM 6 *Find* $S_\alpha \in \mathcal{W}$ *such that*

$$\left(\phi\frac{\partial S_\alpha}{\partial t}, w\right) + \left(\nabla \cdot \mathbf{u}_\alpha(S, \mathbf{u}), w\right) = \left(\frac{q_\alpha(p, S_\alpha)}{\rho_\alpha}, w\right), \quad \forall\ w \in \mathcal{W}$$

In Problem 6, the phase velocities $\mathbf{u}_{\alpha=\mathrm{o,g,w}}$ depends on the total velocity $\mathbf{u}$ and the saturations S as given by equation (6.8).

6.3 Discretization

For spatial discretization, the mixed Finite Element Method (mixed FEM) is used to discretize equations (9.5) and (9.6), whereas the Discontinuous Galerkin Method is used to discretize the conservation law in equation (9.7). For temporal integration the Improved IMPES, [29], is used to decouple the computation of the total velocity and pressure from the computation of the saturations. For simplicity, the forward Euler method is chosen as a time integrator to advance the saturation equations, since the focus of this paper is on the numerical upscaling and to demonstrate scalability in the spatial discretization. While this choice may affect the total numerical efficiency due to the global CFL condition imposed by the explicit time-stepping, AMGe techniques can readily be used together with higher-order and more accurate time discretization schemes.

We stress the fact that all variables and parameters in this section are on the fine grid level and no upscaling is introduced until Section 6.4.

6.3.1 Spatial discretization of the mixed system

The mixed FEM is used to discretize equations (9.5) and (9.6). In particular, we let $\mathcal{R}_h \subset \mathcal{R}$ be the (lowest order) Raviart–Thomas finite element space consisting of vector functions with a continuous normal component across the interfaces between the elements and $\mathcal{W}_h \subset \mathcal{W}$ be the space of piecewise discontinuous polynomials (constant) scalar functions. It is well-known that this choice of finite element spaces satisfies the Ladyzhenskaya-Babuška-Brezzi conditions, and therefore allows for a stable discretization of Problem 11. The Galerkin formulation of the problem reads

PROBLEM 7 *Find* $(\mathbf{u}_h, p_h) \in \mathcal{R}_h \times \mathcal{W}_h$ *such that*

$$\begin{cases} \left(\mathbf{K}_h^{-1}\lambda_h^{-1}\mathbf{u}_h, \mathbf{v}_h\right) - \left(p_h, \nabla\cdot\mathbf{v}_h\right) = \left(g\nabla z_h \sum_\alpha f_{\alpha,h}\rho_{\alpha,h}, \mathbf{v}_h\right), & \forall\ \mathbf{v}_h \in \mathcal{R}_h \\ \left(\nabla\cdot\mathbf{u}_h, w_h\right) - \left(q(p_h), w_h\right) = 0, & \forall\ w_h \in \mathcal{W}_h \end{cases}$$

Here to simplify the notation, we have omitted the dependence of λ_h, $f_{\alpha,h}$ and $q_{\alpha,h}$ on the saturations S_h.

6.3.1.1 Matrix form

Let us denote with $\left\{\boldsymbol{\phi}^j\right\}_{j=1,\ldots,\dim(\mathcal{R}_h)}$ a basis for the space $\mathcal{R}_h$ and $\left\{\psi^j\right\}_{j=1,\ldots,\dim(\mathcal{W}_h)}$ a basis for the space $\mathcal{W}_h$. With this notation, the finite element solution $(\mathbf{u}_h, p_h)$ can be written as a linear combination of the basis functions $(\boldsymbol{\phi}^j, \psi^j)$. More specifically, letting $\mathbf{U} \in \mathbb{R}^{\dim(\mathcal{R}_h)}$ and $P \in \mathbb{R}^{\dim(\mathcal{W}_h)}$ denote the vectors collecting the finite element degrees of freedom $\mathbf{u}_h^i$, $i = 1, \ldots, \dim(\mathcal{R}_h)$ and p_h^i, $i = 1, \ldots, \dim(\mathcal{W}_h)$, we write

$$\mathbf{u}_h = \sum_{j=1}^{\dim(\mathcal{R}_h)} \mathbf{u}_h^j \boldsymbol{\phi}^j, \qquad p_h = \sum_{j=1}^{\dim(\mathcal{W}_h)} p_h^j \psi^j. \tag{6.15}$$

We introduce the finite element matrices M, B whose entries are given by

$$\begin{aligned} M_{ij} = &\quad \left(\mathbf{K}_h^{-1}\lambda_h^{-1}\boldsymbol{\phi}^j, \boldsymbol{\phi}^i\right), && i,j = 1,\ldots,\dim(\mathcal{R}_h) \\ B_{ij} = &\quad \left(\nabla\cdot\boldsymbol{\phi}^j, \psi^i\right), && i = 1,\ldots,\dim(\mathcal{W}_h), j = 1,\ldots,\dim(\mathcal{R}_h). \end{aligned}$$

Finally, we introduce the matrix C that represents the (linear) dependence of the well production rates on the pressure. C is a diagonal semipositive definite matrix with non-zero entries only in the rows corresponding to elements containing a production well. More specifically, we have

$$C_{ij} = \left(\beta\psi^j, \psi^i\right), \qquad i,j = 1,\ldots,\dim(\mathcal{W}_h)$$

where $\beta = \sum_\alpha \mathcal{WI}\lambda_\alpha\chi_{\text{prod}}$ and χ_{prod} is an indicator function with support on the elements that contain a production well.

Problem 9 leads to the solution of the sparse linear system

$$\mathbf{AX} = \mathbf{B}, \tag{6.16}$$

where the block matrix $\mathbf{A}$ and block vectors $\mathbf{X}$ and $\mathbf{B}$ read:

$$\mathbf{A} = \begin{bmatrix} M & B^T \\ B & -C \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{F_u} \\ F_p \end{bmatrix} \tag{6.17}$$

The linear system (7.4) is an indefinite saddle point problem, [20], whose solvability is guaranteed by the fact that M and $C + BB^T$ are symmetric positive definite matrices. If no production wells were present, then $C = 0$ and the pressure would be defined up to a constant.

6.3.2 Spatial discretization of the saturation equations

The conservation law in equation (9.7) is discretized using the Discontinuous Galerkin method. Integration-by-parts is applied to the flux term resulting in

$$\begin{aligned} \Big(\nabla\cdot\mathbf{u}_{\alpha,h}(S_h,\mathbf{u}_h), w_h\Big) = &\sum_{e\in\mathcal{T}_h}\left(-\int_e \mathbf{u}_{\alpha,h}(S_h,\mathbf{u}_h)\cdot\nabla w_h\, d\Omega\right) + \\ &\sum_{e\in\mathcal{T}_h}\sum_{f\in\partial e}\int_f \left(\mathbf{u}_{\alpha,h}(S_h,\mathbf{u}_h)\cdot\mathbf{n}_e\right)^* w_h\, dS \end{aligned} \tag{6.18}$$

where e is an element in the mesh $\mathcal{T}_h$ and f is a face in the set $\mathcal{F}_h$ of all the faces (both boundary and internal faces) in the mesh. The star denotes the numerical

flux $(\mathbf{u}_{\alpha,h}(S_h, \mathbf{u}_h) \cdot \mathbf{n}_e)^*$, which will be specified in the Subsection 6.3.2.1. By introducing the jump notation: $[w_h] = w_h^- - w_h^+$, with the convention depicted in Figure 6.1, we can rewrite (6.18) as

$$\Big(\nabla \cdot \mathbf{u}_{\alpha,h}(S_h, \mathbf{u}_h), w_h\Big) = -\sum_{e \in \mathcal{T}_h} \int_e \mathbf{u}_{\alpha,h}(S_h, \mathbf{u}_h) \cdot \nabla w_h \, d\Omega + \sum_{f \in \mathcal{F}_h} \int_f (\mathbf{u}_{\alpha,h}(S_h, \mathbf{u}_h) \cdot \mathbf{n}_e)^* \, [w_h] \, dS \tag{6.19}$$

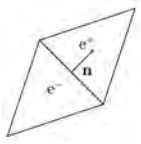


Figure 6.1: Convention used in jump notation.

In the following, we restrict ourselves to the lowest order discretization, which means that $w_h \in \mathcal{W}_h$ is piecewise constant on elements, and therefore Problem 6 reduces to

$$\int_\Omega \phi \frac{\partial S_{\alpha,h}}{\partial t} w_h \, d\Omega + \sum_{f \in \mathcal{F}_h} \int_f (\mathbf{u}_{\alpha,h}(S_h, \mathbf{u}_h) \cdot \mathbf{n}_e)^* \, [w_h] \, dS = \int_\Omega \frac{q_{\alpha,h}}{\rho_\alpha} w_h \, d\Omega, \tag{6.20}$$

In matrix form, we have

$$\phi W \frac{\partial S_{\alpha,h}}{\partial t} = F(S_h, \mathbf{u}_h) + \frac{1}{\rho_\alpha} W q_{\alpha,h} \tag{6.21}$$

where

$$F(S_h, \mathbf{u}_h) = \sum_{f \in \mathcal{F}_h} \int_f (\mathbf{u}_{\alpha,h}(S_h, \mathbf{u}_h) \cdot \mathbf{n}_e)^* \, [w_h] \, dS, \tag{6.22}$$

and W is a mass matrix corresponding to the L^2-inner products for functions in $\mathcal{W}_h$. In the lowest order case, W reduces to a diagonal matrix, whose entry (i, i) represents the volume of element i.

6.3.2.1 Choice of the numerical fluxes

In this section, we describe how the numerical fluxes in (6.20) are computed.

To this aim, we need to evaluate the restriction of the phase velocity to the faces of the elements in the mesh. The phase velocity is defined by following expressions

$$\mathbf{u}_{\alpha,h} = f_{\alpha,h}\left(\mathbf{u}_h + \mathbf{G}_{\alpha,h}\right), \quad \mathbf{G}_{\alpha,h} = \sum_{\substack{\beta=o,w,g \\ \beta\neq\alpha}} \lambda_{\beta,h}(\rho_\alpha - \rho_\beta)\mathbf{K}_h g \nabla z, \tag{6.23}$$

where $\lambda_{\beta,h}$, ρ_α, $\mathbf{K}_h$ and $f_{\alpha,h}$ are piecewise constants associated with each element. Because of the fluid incompressibility assumption, ρ_α and ρ_β are constant over the domain and therefore their restrictions to the faces of the mesh are trivially defined. To evaluate the permeability on a face f we follow the approach presented in [30, p. 134] and compute a component-wise harmonic mean of the values of the permeability tensor coming from element + and element − sharing face f

$$(\mathbf{K}_h)^f_{i,j} = \frac{2(\mathbf{K}_h)^+_{i,j}\,(\mathbf{K}_h)^-_{i,j}}{(\mathbf{K}_h)^+_{i,j} + (\mathbf{K}_h)^-_{i,j}}, \tag{6.24}$$

where i, j denote the components in the tensor. However, this procedure is only an approximation and may impact the gravity term (i.e. the vertical component) for distorted, sloping elements and full tensor coefficients. Other approaches could also be used to approximate the restriction of $\mathbf{K}$ to a face of the mesh such as upwinding $\mathbf{K}$ in the direction of the phase velocity (see [13]).

The mobility $\lambda_{\beta,h}$ and the fractional flow function $f_{\alpha,h}$ in equation (6.23), which are part of the flux in equation (6.22), are approximated when reconstructing the flux across faces using an upwind method. The numerical fluxes are upwinded in the normal direction of the interfaces according to the flow direction of the individual phases

$$\Big(\mathbf{u}_{\alpha,h}(S_h, \mathbf{u}_h)\cdot\mathbf{n}\Big)^* = \begin{cases} \mathbf{u}_{\alpha,h}(S_h^-, \mathbf{u}_h^-)\cdot\mathbf{n} & \text{if} \quad (\mathbf{u}_{\alpha,h}\cdot\mathbf{n})^* \geq 0 \\ \mathbf{u}_{\alpha,h}(S_h^+, \mathbf{u}_h^+)\cdot\mathbf{n} & \text{if} \quad (\mathbf{u}_{\alpha,h}\cdot\mathbf{n})^* < 0. \end{cases} \tag{6.25}$$

Without upwinding, the numerical solution may display oscillations, overshoots or undershoots (e.g., saturations less than zero or greater than one), or converge to an incorrect solution, [16, p.163].

Equation (6.25) only implicitly defines the numerical fluxes $\mathbf{u}^*_{\alpha,h}$, since the upwind direction depends on both the total velocity and the gravitational term $\mathbf{G}_{\alpha,h}$. Without gravity the direction of the phase velocity is the same as the direction of the total velocity, i.e.

$$\Big(\mathbf{u}_{\alpha,h}(S_h, \mathbf{u}_h)\cdot\mathbf{n}\Big)^* = \begin{cases} \mathbf{u}_{\alpha,h}(S_h^-, \mathbf{u}_h^-)\cdot\mathbf{n} & \text{if} \quad (\mathbf{u}_h\cdot\mathbf{n}) \geq 0 \\ \mathbf{u}_{\alpha,h}(S_h^+, \mathbf{u}_h^+)\cdot\mathbf{n} & \text{if} \quad (\mathbf{u}_h\cdot\mathbf{n}) < 0. \end{cases} \tag{6.26}$$

When accounting for gravity, the $\mathbf{G}_{\alpha,h}$ term also needs to be considered when finding the upwind direction. Hence determining the direction of the phase velocities is non-trivial, since $\mathbf{G}_{\alpha,h}$ actually contains one of the properties: $\lambda_{\beta,h}$

that we are trying to approximate in the upwinding. In our simulator, to determine phase velocity directions and upwinding, we use the approach proposed in [88] (see also [85]), which uses a heuristic approach to determine the upwind direction for $\mathbf{G}_{\alpha,h}$, and then check for consistency of the phase velocities $\mathbf{u}_{\alpha,h}$ and the upwind direction used for $\mathbf{G}_{\alpha,h}$.

6.3.3 Temporal discretization

Only the conservation law in equation (6.21) has an explicit time derivative dependence. The discretized system of equations is solved using an "IMplicit Pressure Explicit Saturations" (IMPES) type method or – more accurately for a mixed system – "IMplicit Pressure And Velocity Explicit Saturations". Specifically an improved IMPES type method, [29], is employed, where sub-time stepping is used for saturations. Improved IMPES can be written as a fractional step time-advancing technique, where a large time step ΔT is used to update the total velocity and pressure unknowns, and a smaller time step $\Delta t = \frac{\Delta T}{k}$ is used to update the saturation unknowns. By denoting with $S^{n+\frac{i}{k}}$ the saturations at time $t = n\Delta T + i\Delta t$, we write the fractional step saturation update as

$$S_{\alpha,h}^{n+\frac{i+1}{k}} = S_{\alpha,h}^{n+\frac{i}{k}} + \Delta t \frac{1}{\phi} W^{-1}(F(S_h^{n+\frac{i}{k}}, \mathbf{u}_h^n) + \frac{1}{\rho_\alpha} W q_{\alpha,h}(S_h^{n+\frac{i}{k}}, p_h^n)), \quad (6.27)$$

where $\mathbf{u}_h^n$ and p_h^n are the velocity and pressure at time $t = n\Delta T$ computed by solving Problem 9.

A pseudo-code for the implementation is listed in Algorithm 5.

Algorithm 3 - Pseudo code for improved IMPES implementation.

```
while t < t_final do
    Assemble mixed system in Problem 9 given current saturations S_h^n
    Solve the mixed system (7.4) for pressure p_h^n and total velocity u_h^n
    Choose Δt based on CFL condition in equation (6.29) and let k = ΔT/Δt
    for 0, ..., k − 1 do
        Compute S_h^{n+(i+1)/k} with equation (6.27)
        t = t + Δt
    end for
end while
```

Here $q_{\alpha,h}$ is computed as

$$q_{\alpha,h}(S_h, p_h) = \begin{cases} rate & \text{if injector} \\ \rho_\alpha f_{\alpha,h}(S_h) q_h(p_h) & \text{if producer.} \end{cases} \quad (6.28)$$

where $q_h(p_h)$ is the total injection/production rate defined by equation (6.5), and *rate* is a user-supplied input.

To ensure the stability of the discretization, we choose Δt such that

$$\Delta t \leq \frac{c}{\phi} \min \left(\frac{h}{\left\| \frac{\partial \mathcal{F}(S, \mathbf{u})}{\partial S} \right\|_\infty}, \frac{\rho_\alpha}{\left\| \frac{\partial q_\alpha(S, \mathbf{u})}{\partial S} \right\|_\infty} \right), \tag{6.29}$$

where c is a user supplied real number between 0 and 1 (in our tests $c = 0.9$). This choice is motivated by the fact that the first constraint for the time step is the CFL condition for the pure transport equation

$$\phi \frac{\partial S}{\partial t} = \nabla \cdot \mathcal{F}(S, \mathbf{u}) \tag{6.30}$$

and the second term is dictated by the stability region of forward Euler for the ODE

$$\phi \frac{\partial S}{\partial t} = \frac{1}{\rho_\alpha} q_\alpha(S, p). \tag{6.31}$$

6.4 Element-based Algebraic Multigrid (AMGe)

AMGe is a framework for multigrid methods tailored to systems stemming from finite element discretizations. The components in AMGe are constructed from local element information, such as finite element matrices and element topology. This is in contrast to classical AMG, where only system coefficients are used to construct the hierarchy of coarse spaces. It should be noted that the AMGe framework is significantly different from (the classical) AMG. Specifically, with this particular version of AMGe, we have guaranteed approximation properties, where a good convergent two-level AMG only has a weak approximation property (as a necessary condition, cf., the survey [115]). The latter deteriorates as the coarsening ratio (H/h) increases. This means that in the coarsening procedure, AMGe does not depend on a good heuristic as used in AMG. That said, the choice of agglomerated elements does affect the resulting coarse spaces and in the numerical results in Section 8.6, it is demonstrated that taking into account the mesh anisotropy in the construction of the coarse agglomerated meshes improves the accuracy of the coarse systems.

Following the approach first described in [73, 72], an overview of the techniques involved for this specific version of AMGe are described in this section. For a more detailed description and for fundamental theory of the properties of the

method, see [73] and [99]. The technique introduced in [73] extends the approach described in [99] and guarantees approximation properties of the coarse spaces for general unstructured meshes. The improved approximation properties of the coarse velocity spaces are achieved by introducing additional degrees of freedom associated with non-planar interfaces between agglomerates. This leads to coarse spaces with the same stability and approximation properties as of the original (fine-grid) Raviart-Thomas space.

6.4.1 Constructing agglomerates

Agglomerates are formed by grouping together fine-grid elements. Different techniques are available in our framework: graph partitioning techniques, octrees, geometric (coordinate-based) mesh partitioners, and also other techniques that exploit directly the cartesian or refinement structure of the mesh. To construct agglomerated elements using graph partitioning techniques, in particular, we build the dual graph of the mesh, which is an undirected graph, where each node of the graph represents an element in the mesh and node i is connected to node j if element i and element j share a face. METIS, [66], is used for the partitioning of the undirected graph resulting in agglomerates consisting of fine-grid elements. Weights of the nodes and links in the dual graph (i.e. for the elements and faces of the mesh) can be provided to the partitioners in order to generate smaller agglomerated elements in parts of the domain or to modify the aspect ratio of coarse elements (see Section 6.5.2). The coarse faces consist of the fine faces belonging to the intersection of any pair of neighboring agglomerates. This means that for unstructured meshes the coarse faces are non-planar in most cases. The tentative number of fine-grid elements per agglomerate is a user-given input. A possible way to further improve the quality of the agglomerated meshes is by using problem-dependent weights for the graph partitioners [89].

It was found that straightforward graph partitioning algorithms sometimes produce agglomerates with "bad" or undesired topological properties, such as tunnels or holes. Breaking up the "bad" coarse elements into smaller agglomerates is then necessary in order to guarantee the well-posedness of the local problems involved in the computation of the coarse spaces (see Section 6.4.3).

6.4.2 Building coarse pressure spaces

The coarse pressure spaces are constructed in the same way as introduced in [99], where the coarse space consists of piecewise constant functions on agglom-

erated elements. We define the coarse pressure space $\mathcal{W}_H \subset \mathcal{W}_h$ to consist of functions which are constant on each agglomerated element. In addition, one can enrich the space $\mathcal{W}_H$ by restricting additional functions, such as (higher order) polynomial functions or other functions of interest. It is worth noticing that the coarse pressure space does not need to be conforming across agglomerated element interfaces. A basis of the space $\mathcal{W}_H$ is then used to form the columns of the prolongation matrix $\mathbf{P}_p : \mathcal{W}_H \to \mathcal{W}_h$. In the following section, we will demonstrate how to construct a coarse velocity space $\mathcal{R}_H \subset \mathcal{R}_h$ such that $\nabla \cdot \mathcal{R}_H = \mathcal{W}_H$. This property is necessary to preserve the stability of the upscaled discretization and to guarantee that the spaces $(\mathcal{R}_H, \mathcal{W}_H)$ are inf-sup compatible.

It should be noted that in our coarsening procedure defined in section 6.4.5, p_H is a Galerkin projection of p_h via the solution of a coarse system (in the linear case).

Furthermore, we will assume that also the saturations are upscaled using the same coarse space $\mathcal{W}_H$. This is consistent with the choice of finite element spaces for the fine grid discretization, but one could use different coarse spaces for upscaling the saturations.

6.4.3 Building coarse velocity spaces

The method used to create coarse finite element spaces for the lowest order Raviart-Thomas space is described here. The explanation closely follows that of [73]. The method is a two-step process, where we first find the coarse basis functions on coarse faces and then extend the basis functions into the interior of the neighboring agglomerated elements.

The coarse basis functions are defined in terms of their fine degrees of freedom. Given a sufficiently smooth vector function $\mathbf{r}$ and a fine face f, the value of the degree of freedom associated with f is defined as

$$\mathrm{DoF}_f(\mathbf{r}) = \int_f \mathbf{r} \cdot \mathbf{n}_f \, dA, \tag{6.32}$$

where $\mathbf{n}_f$ is the unit normal to the fine face and A is the surface area of the face.

For each coarse face F, a matrix $\mathbf{W}_F$ is formed. $\mathbf{W}_F$ consists of the values of

the DoF of the fine faces $f_1, \ldots, f_{|F|}$ constituting F

$$\mathbf{W}_F = \begin{bmatrix} \mathrm{DoF}_{f_1}(\mathbf{e}_1) & \mathrm{DoF}_{f_1}(\mathbf{e}_2) & \mathrm{DoF}_{f_1}(\mathbf{e}_3) & \mathrm{sgn}(f_1, F) \int\limits_{f_1} dA \\ \vdots & \vdots & \vdots & \vdots \\ \mathrm{DoF}_{f_{|F|}}(\mathbf{e}_1) & \mathrm{DoF}_{f_{|F|}}(\mathbf{e}_2) & \mathrm{DoF}_{f_{|F|}}(\mathbf{e}_3) & \mathrm{sgn}(f_{|F|}, F) \int\limits_{f_{|F|}} dA \end{bmatrix}, \tag{6.33}$$

where $|F|$ is the number of fine faces in the coarse face F. Here $\mathrm{sgn}(f, F) = 1$ if the orientation of the fine face is equal to that of the coarse face and $\mathrm{sgn}(f, F) = -1$ otherwise. Above, $\mathbf{e}_i$ stand for the three coordinate constant vector-functions. The goal is to ensure that the coarse Raviart-Thomas space contains locally (on each agglomerated element) these constant vectors, hence have first order of approximation in L^2 as the fine-grid Raviart-Thomas space. The above construction is fairly general; we can include any given velocity functions of our interest, in the coarse velocity space and maintain the compatibility, i.e., the coarse pressure space should contain their divergence.

Using an SVD decomposition $\mathbf{W}_F = \mathbf{U}\Sigma\mathbf{V}^T$, the linearly dependent columns of $\mathbf{W}_F$ are eliminated. It is worth noticing that the cost of computing such SVD scales linearly with the number of fine degree of freedom which belongs to the coarse face. This is due to the fact that (1), we perform the *thin* SVD [48] and (2), that the number of columns in $\mathbf{W}_F$ is small and independent of the number of fine degree of freedom. The left singular vectors (columns of $\mathbf{U}$) $\mathbf{u}_j$ are chosen based on the corresponding singular values σ_j. If $\sigma_j \geq \epsilon\sigma_{\max}$, where $\epsilon \in (0, 1]$ is a user-given input, then $\mathbf{u}_j$ defines a coarse basis function for the coarse face F denoted $\mathbf{r}_F^j$. More precisely, $\mathbf{r}_F^j$ is only equal to $\mathbf{u}_j$ on the coarse face F and zero everywhere else. If the coarse face F is planar, then only σ_1 would be different from 0 and only one coarse trace will be selected, for a non-planar face up to four coarse traces will be selected depending on the tolerance ϵ. This is in contrast with multilevel multiscale mimetic methods (M^3), [76, 75], where only one coarse degree of freedom is used for each face.

The above procedure describes the first step to finding a coarse velocity space. The second step involves taking the partially defined functions $\mathbf{r}_F^j$ and extending these coarse basis functions into the interior of the coarse elements. The extension is performed using the approach in [99], which guarantees that the divergence of the coarse Raviart-Thomas space belongs to the coarse L^2 space. More specifically, for each coarse element T we define the local finite element spaces:

$$\widetilde{\mathcal{R}}_T = \{\mathbf{v}_h \in \mathcal{R}_h \,|\, \mathrm{supp}(\mathbf{v}_h) \subset T \,\mathrm{and}\, \mathbf{v}_h \cdot \mathbf{n} = 0 \,\mathrm{on}\, \partial T\},$$
$$\text{and} \quad \widetilde{\mathcal{W}}_T = \{w_h \in \mathcal{W}_h \,|\, \mathrm{supp}(w_h) \subset T \,\mathrm{and}\, (w_h, 1) = 0\}.$$

Given a partially defined function $\mathbf{r}_F^j$ on the coarse face F belonging to the coarse element T, the local (element-based) mixed system reads

PROBLEM 8 *Find* $(\mathring{\mathbf{r}}_T, p_h) \in \widetilde{\mathcal{R}}_T \times \widetilde{\mathcal{W}}_T$ *such that*

$$\begin{cases} \left(\alpha\,(\mathring{\mathbf{r}}_T + \mathbf{r}_F^j), \mathbf{v}_h\right)_T + (p_h, \nabla\cdot\mathbf{v}_h)_T = 0, & \forall\ \mathbf{v}_h \in \widetilde{\mathcal{R}}_T \\ \left(\nabla\cdot(\mathring{\mathbf{r}}_T + \mathbf{r}_F^j), w_h\right)_T = 0, & \forall\ w_h \in \widetilde{\mathcal{W}}_T \end{cases}$$

The coefficient matrix α (a 3×3 SPD matrix) can be set equal to the coefficients from the original problem, $\mathbf{K}^{-1}$, but this is not strictly necessary. Problem 8 is guaranteed to have a unique solution, [73, 99]. By solving these local problems on each pair of agglomerates (T^+, T^-) adjacent to a coarse face F, we obtain the coarse basis functions $\mathbf{r}_h = \mathbf{r}_F^j + \mathring{\mathbf{r}}_{T^+} + \mathring{\mathbf{r}}_{T^-}$ of the space $\mathcal{R}_H$. We finally let the columns of the prolongation matrix $P_{\mathbf{u}} : \mathcal{R}_H \to \mathcal{R}_h$ be the collection of the coarse basis functions $\mathbf{r}_h$.

6.4.4 Dealing with localized sources

Localized or point-wise source terms, such as the well's terms in equation (9), represent an additional challenge for many upscaling techniques. In fact, localized source terms may drastically reduce the accuracy of traditional upscaling techniques based on homogenization, since the exact location of the source is lost on the coarser mesh.

In contrast, the AMGe approach offers great flexibility in dealing with localized sources and it allows for accurate upscaling of localized sources. In particular, two different approaches are possible. One option is to use smaller agglomerated elements in proximity of a localized source. This can be achieved by simply leaving some fine elements in the neighborhood of the localized source unagglomerated, or by using a weighted graph partitioning algorithm. This approach is the equivalent of adaptive mesh refinement (h-refinement) in the finite element settings. The other option, is to locally enrich the coarse space by adding additional functions with support in a neighborhood of the localized sources, as describe above. This approach is similar in spirit to p-refinement in the finite element settings. Actually, in our case these approaches should be viewed as *adaptive coarsening*. In the numerical results presented in this paper we use the first approach to locally refine the agglomerated mesh in the neighborhood of a localized source. Figure 6.12c provides an example of an agglomerated mesh where the elements which contain wells and their immediate neighbor cells are left unagglomerated.

6.4.5 Upscaling with AMGe

The construction of the coarse spaces and thereby the interpolation operators are done in a setup phase, while the rest of the computation is entirely performed on the coarse agglomerated meshes. The fine grid is visited only when the solution is prolongated back to the fine grid for visualization purposes. For some applications, where there is no need for the solution on the fine grid, but only scalar quantities – such as production data – are important, the fine grid is therefore touched only once (in the setup phase) and not during the rest of the simulation.

The upscaled mixed system,

$$\begin{bmatrix} M & B^T \\ B & -C \end{bmatrix}_H \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix}_H = \begin{bmatrix} \mathbf{F_u} \\ F_p \end{bmatrix}_H, \tag{6.34}$$

is assembled directly for the upscaled spaces. The local coarse matrices (one for each agglomerated element) are precomputed in the set-up phase and then assembled into the upscaled system at each time step without visiting the fine grid. By construction, the upscaled mixed system assembled on the coarse spaces is equivalent to the following Galerkin projection

$$\begin{bmatrix} M & B^T \\ B & -C \end{bmatrix}_H = \begin{bmatrix} P_u^T & 0 \\ 0 & P_p^T \end{bmatrix} \begin{bmatrix} M & B^T \\ B & -C \end{bmatrix}_h \begin{bmatrix} P_u & 0 \\ 0 & P_p \end{bmatrix}, \begin{bmatrix} \mathbf{F_u} \\ F_p \end{bmatrix}_H = \begin{bmatrix} P_u^T & 0 \\ 0 & P_p^T \end{bmatrix} \begin{bmatrix} \mathbf{F_u} \\ F_p \end{bmatrix}_h, \tag{6.35}$$

where H and h, in this case, respectively represents the upscaled level and the fine grid level. The strong conservation properties of the finite element spaces ($\mathcal{R}_h$, $\mathcal{W}_h$), the AMGe coarsening technique that ensures $\nabla \cdot \mathcal{R}_H = \mathcal{W}_H$ (described in Section 4.3) and the special treatment of the wells (described in Section 4.5) guarantee a strong mass conservation for the upscaled system. In particular, when we leave wells unagglomerated, we have that $f_h = P_p f_H$ and therefore $(\nabla \cdot u_H - C_H p_H, w_H) = (f_H, w_H)$ implies $(\nabla \cdot P_u u_H - C_h P_p p_H, w_h) = (f_h, w_h)$.

In a similar way as the mixed system, the upscaled saturation equations read

$$\int_\Omega \phi \frac{\partial S_{\alpha,H}}{\partial t} w_H \, d\Omega + \sum_{F \in \mathcal{F}_H} \sum_{f \in \mathcal{F}} \int_f (\mathbf{u}_{\alpha,H}(S_H, \mathbf{u}_H) \cdot \mathbf{n})^* \, [w_H] \, dS = \int_\Omega \frac{q_{\alpha,H}}{\rho_\alpha} w_H \, d\Omega,$$

where $\mathcal{F}_H$ represents the set of coarse faces F in the agglomerated set. It is worth to notice that for the choice of piecewise constant saturations on agglomerated elements the local matrices representing the integral $\sum_{f \in \mathcal{F}} \int_f (\mathbf{u}_{\alpha,H}(S_H, \mathbf{u}_H) \cdot \mathbf{n})^* \, [w_H] \, dS$ can be precomputed in the set-up phase for each coarse degree of

Porosity ϕ	0.3	-
Viscosity oil μ_o	1.14	cP
Viscosity water μ_w	0.096	cP
Density oil ρ_o	800	kg/m^3
Density water ρ_w	1022	kg/m^3

Table 6.1: Input parameters.

freedom of the coarse total velocity $\mathbf{u}_H$. Therefore to evolve the saturation equations in time is not necessary to visit the fine mesh during the simulation.

In the numerical results presented in the following section, we use MINRES preconditioned by a block-diagonal AMG preconditioner to solve the mixed system involving total velocity and pressure (see [81] or [114]). The upscaled coarse grid problems are solved with a sparse direct solver. For large-scale problems, the coarse grid problems should also be solved with an iterative solver. However, the coarse grid mixed systems can be more ill-conditioned than the original fine grid system due to the complicated geometry of the agglomerates and the possibly non-uniform distribution of coarse degrees of freedom within each agglomerated element. This issue is not addressed in the present paper and we refer to [77] which deals with our progress on the construction of efficient solvers for the upscaled system using iterative methods.

6.5 Numerical results

Two numerical experiments are carried out to test the accuracy of the numerical upscaling. We consider the 10th SPE Comparative Solution Project (SPE10 dataset 2) top layer [95] and a 3D model derived from the SAIGUP model where some features (i.e. faults) were not included. The SPE10 model has a regularly structured mesh with a highly heterogeneous permeability field. The SAIGUP model has a more challenging and realistic geometry. Simulations are carried out using our numerical upscaling technique and the results are compared to the simulation results from the fine grid reference model. For all simulations, the input values given in Table 6.1 are used.

The software developed in this work uses the finite element library MFEM, [1], from Lawrence Livermore National Laboratory (LLNL). MFEM is a general, modular, parallel C++ library for finite element methods research and development. It supports a wide variety of finite element spaces in 2D and 3D, as well as many bilinear and linear forms defined on them. It includes classes for dealing with various types of triangular, quadrilateral, tetrahedral and hexahe-

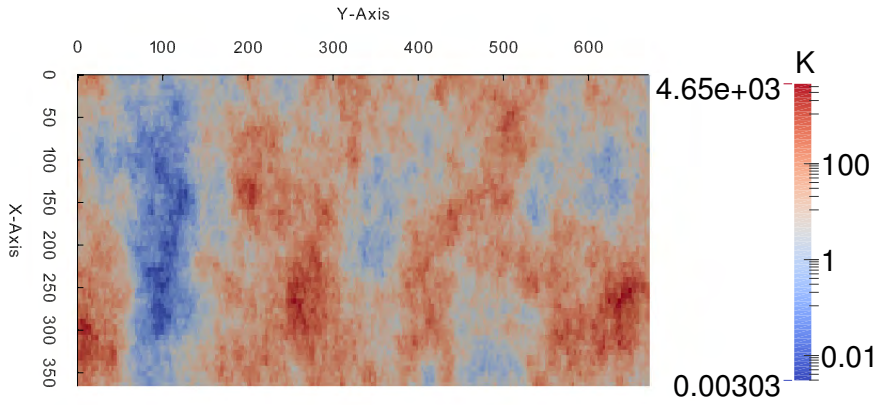


Figure 6.2: SPE10 x-permeability for the top layer.

dral meshes and their global and local refinement. Parallelization in MFEM is based on MPI, and it leads to high scalability in the finite element assembly procedure. It supports several solvers from the hypre library, [53].

6.5.1 SPE10 top layer

As a first example, a test case using the top layer of the SPE10 x-permeability field, is studied, [95]. A 2D model is chosen as a first case to allow for better analyses of the accuracy obtained in the upscaling procedure. Figure 6.2 displays the permeability field. We simulate the fine grid reference model and compare this to 3 upscaled models with different levels of coarsening.

The mesh is a regularly structured grid with $60 \times 220 \times 1$ elements, where each element has the size: $6.096 \times 3.048 \times 0.6096$ meters. A water injection well is placed in the middle of the mesh and 4 producers are placed in the corner elements. The producers are controlled by a bottom hole pressure of 175 bar. The injection well is controlled by a constant injection rate of 0.5 times the element volume. All wells have a radius of 0.2 meters. The porosity is 0.3 for all elements. The whole reservoir has an initial oil saturation of 1.

Constant time steps of 10 days are used. The simulation horizon is 10 years. Time integration is carried out with the Improved IMPES method as explained in Section 6.3.3. Since the main interest of the paper is on the numerical upscaling, we have purposely kept the time stepping very simple. Figures 6.3 and 6.4 show the daily and accumulated production (and accumulated injection)

Problem	#elements	#faces	#DoFs	nnz	arithmetic complexity	operator complexity
60 × 220 (fine)	13200	53080	92680	607485	-	-
30 × 110	3317	13418	23371	152965	1.25252	1.2518
15 × 55	842	3446	5986	39327	1.06491	1.06474
6 × 22	149	634	1093	7351	1.01199	1.0121

Table 6.2: Degrees of freedom, number of non-zeros and complexities.

for both the fine grid reference solution and the upscaled solutions. Structured coarsening is used, where respectively 4 (30×110), 16 (15×55) and 100 (6×22) fine grid elements are grouped together into one agglomerate. Table 7.1 contains information about the #DoFs, number of non-zeros and complexities for the different levels of coarsening. The arithmetic complexity C_a is defined as the ratio of the total number of degrees of freedom on all levels (fine grid and upscaled) to the fine grid number of degrees of freedom. In a similar way, the operator complexity C_o is the ratio of the total number of non-zeros (in the mixed system) on all levels to the number of non-zeros on the fine grid. More specifically, we have

$$C_a = \frac{\sum_{l=0}^{\text{levels-1}} \dim(\mathcal{R}_{h(l)} \times \mathcal{W}_{h(l)})}{\dim(\mathcal{R}_{h(0)} \times \mathcal{W}_{h(0)})} \qquad C_o = \frac{\sum_{l=0}^{\text{levels-1}} \text{nnz}(\mathbf{A}_{h(l)})}{\text{nnz}(\mathbf{A}_{h(0)})}. \tag{6.36}$$

We stress upon the fact that many methods in practice can achieve C_a close to unity and have acceptable approximation properties. However, it is also of vital practical importance to also ensure that C_o is close to unity (or at least sufficiently less than two) since then we can store the upscaled problem with memory (much) less than the original fine-grid problem.

As it can be seen from Figures 6.3 and 6.4, the difference between the fine grid reference solution and the upscaled solutions is small and even difficult to spot in these plots.

To get a better idea of the errors committed in the upscaling, Figures 7.3 and 6.6 show the difference between the upscaled and the fine grid production relative to the fine grid production/injection. Both daily and accumulated production error curves are shown. As evident from the figures, the error committed even for highly aggressive coarsening (100 fine elements per agglomerate) is less than 3%.

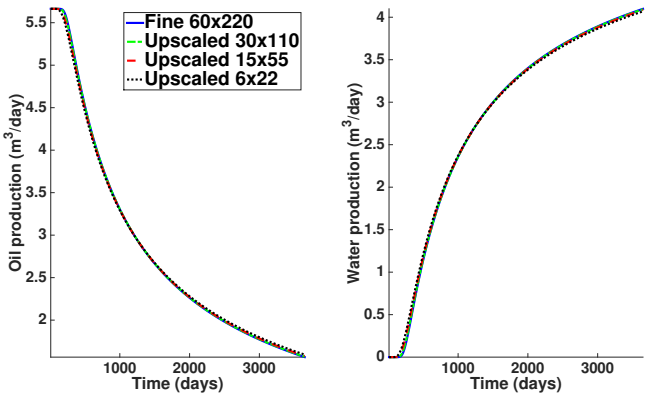


Figure 6.3: Daily production data for fine grid 60×220 and upscaled models.

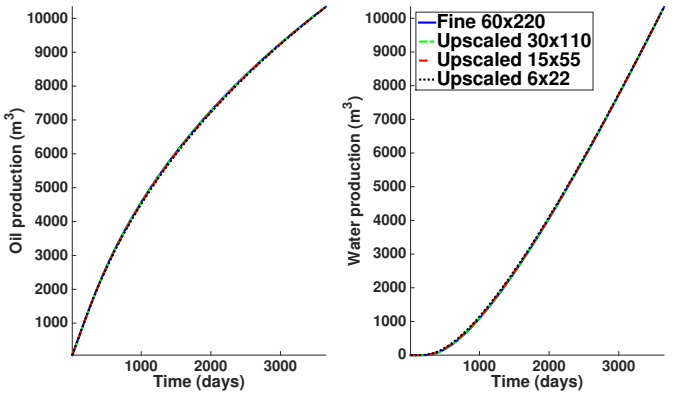


Figure 6.4: Accumulated production data for fine grid 60×220 and upscaled models.

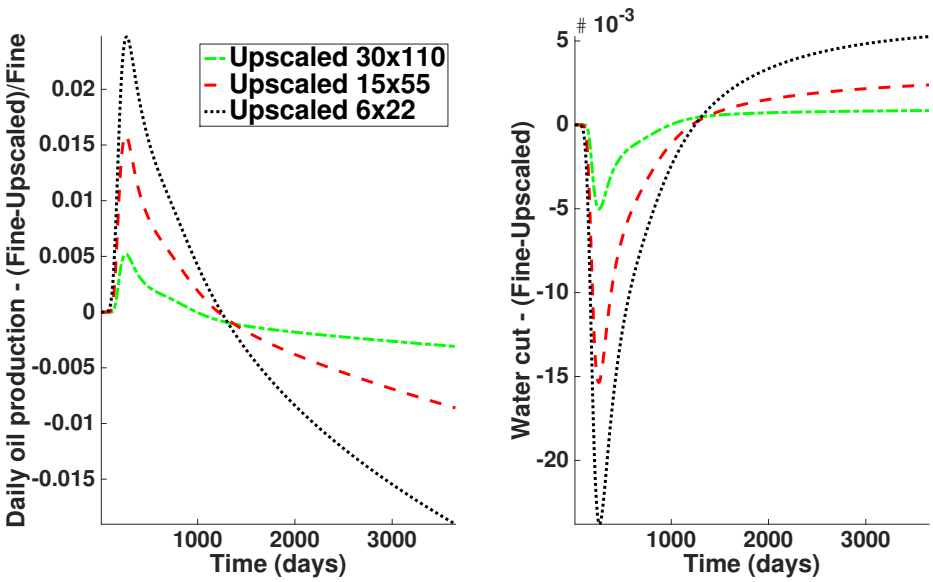


Figure 6.5: Difference between fine grid 60×220 and upscaled models in terms of daily oil production and water cut.

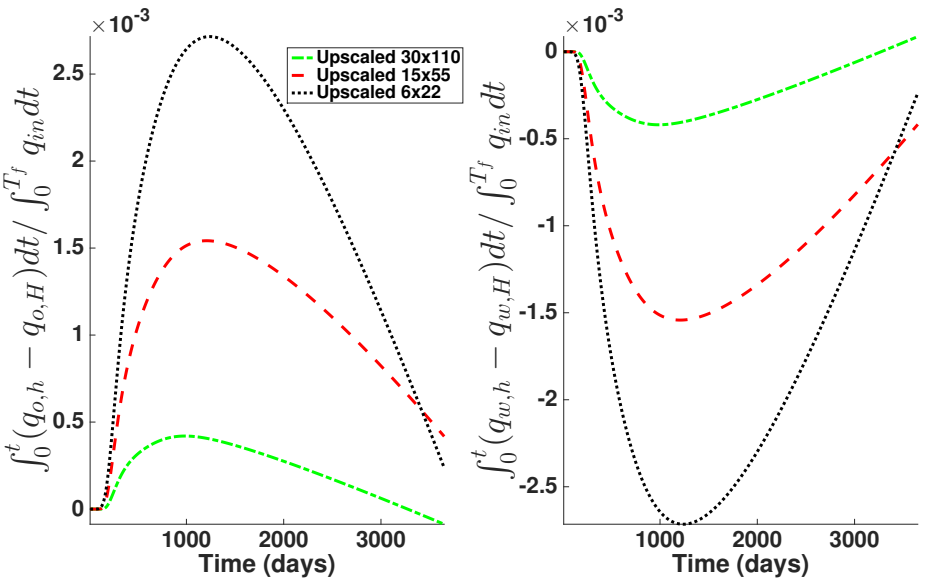


Figure 6.6: Difference in production between the fine grid and upscaled models integrated over time relative to the total injection of water over time.

By comparing Figure 6.3 and Figure 7.3, it is clear that most of the error is at the time where water arrives at the production wells. It seems that the upscaled models predict slightly earlier water breakthrough compared to the fine grid reference solution. This is to be expected, since the upscaling results in a more diffusive discretization, which means small amounts of water will be moved more quickly to the production wells. Mitigating this effect is an issue, which will be covered in future communications, e.g. by adaptively refining the coarse spaces in the neighborhood of the water front. Nevertheless, the error committed in the upscaling is very small and it is a very satisfying result for such a heterogeneous permeability field, where permeability values are varying six orders of magnitude.

As previously mentioned, the upscaled solutions have a tendency to predict water breakthrough at an earlier time compared to the fine grid reference solution. To better estimate exactly how much earlier, Table 6.3 contains the number of days before water breakthrough is predicted at each well for both the fine grid solution and the upscaled solutions. In this table water breakthrough is defined as the time where 1% of the produced fluids is water.

	Coarse problem size			
Well	60×220	30×110	15×55	6×22
Bottom right	210	200	170	160
Top right	490	470	410	330
Top left	810	820	830	810
Bottom left	680	680	650	610

Table 6.3: Number of days before water breakthrough for each well.

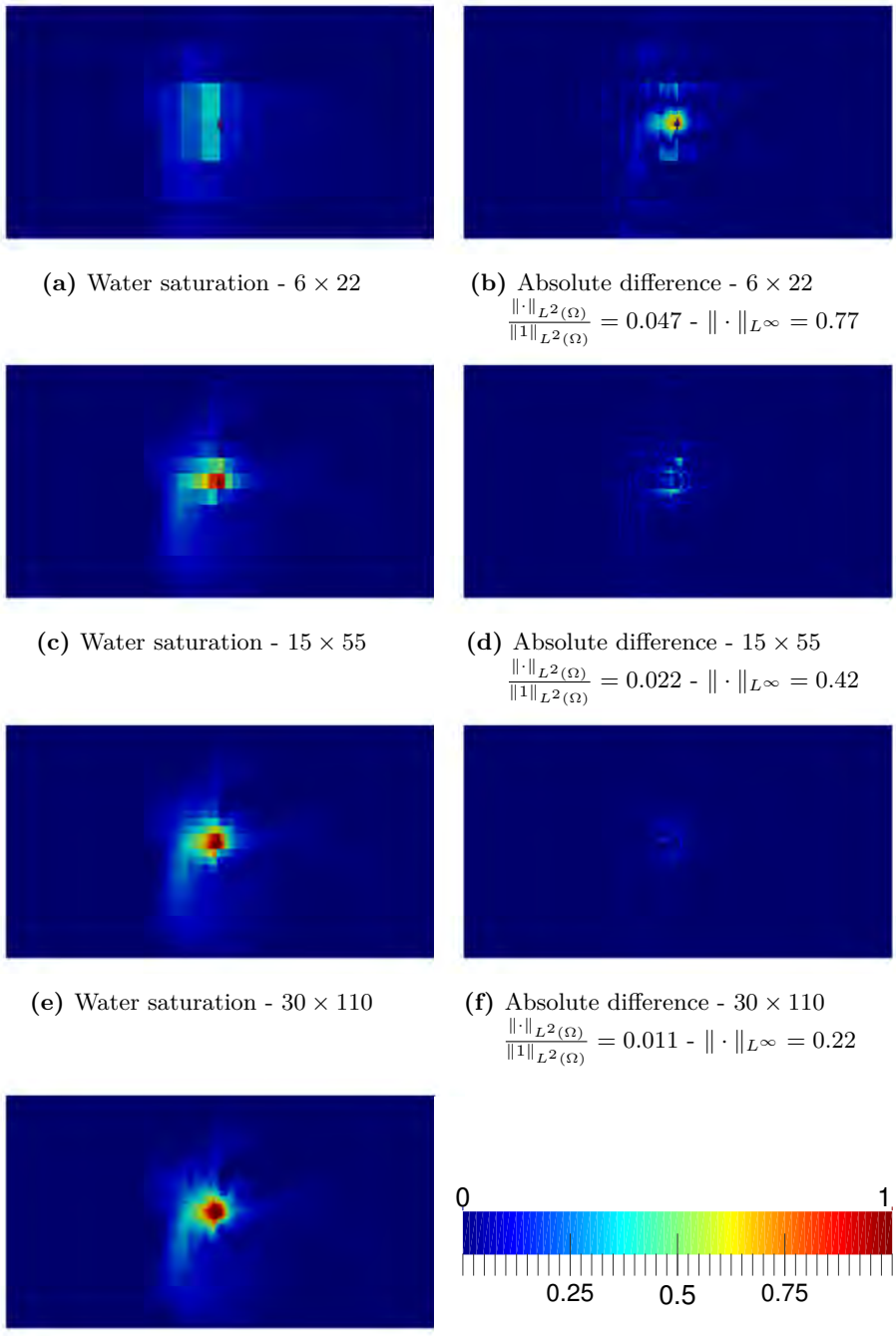
Clearly, the amount of coarsening has a large impact on the ability to accurately predict water breakthrough. The upscaled solutions seem to compensate for the early water breakthrough by later producing less water, ultimately resulting in the accumulated production to converge over time towards the same amount. This compensating behavior was also observed using other upscaling techniques that enforce strong mass conservation and in absence of capillary pressure (see e.g [118]).

In order to give a better picture of how the upscaled saturation profiles compare to the fine grid saturation profile, the water saturation at the time of water breakthrough and after 10 years is plotted in Figures 6.7 and 6.8. From the top and down, the plots show the upscaled 6×22, 15×55, 30×110 together with the the fine grid 60×220 water saturations. The absolute difference between the fine grid reference solution and the upscaled solutions is plotted in the right column. The errors given in the captions are computed by interpolating the upscaled solution onto the fine grid, subtracting this from the fine grid reference solution and then computing the weighted $L^2(\Omega_h)$ inner product.

6.5.1.1 Multilevel recursive upscaling

In this section, the multilevel recursive upscaling capability of the AMGe method is demonstrated for the top layer of the SPE10. Figure 6.9 shows the agglomeration of the fine grid elements. METIS is used in a recursive way to find successive coarse levels of agglomerated elements. Five levels (level zero is the fine grid mesh) are generated, where a coarsening factor of sixteen is used from level zero to level one and a coarsening factor of four for the remaining agglomeration steps. Elements containing wells are left unagglomerated. Given this hierarchy of agglomerated elements, simulations are carried out on each level and the resulting upscaled solutions are compared to the fine grid reference solution.

Figure 6.10 shows the error in the solution between the fine grid reference solution (level zero) and the solutions for the four (level 1,2,3,4) upscaled problems as a function of h/H (approximated by the square root of the ratio of the num-



(a) Water saturation - 6×22

(b) Absolute difference - 6×22
$\frac{\|\cdot\|_{L^2(\Omega)}}{\|1\|_{L^2(\Omega)}} = 0.047$ - $\|\cdot\|_{L^\infty} = 0.77$

(c) Water saturation - 15×55

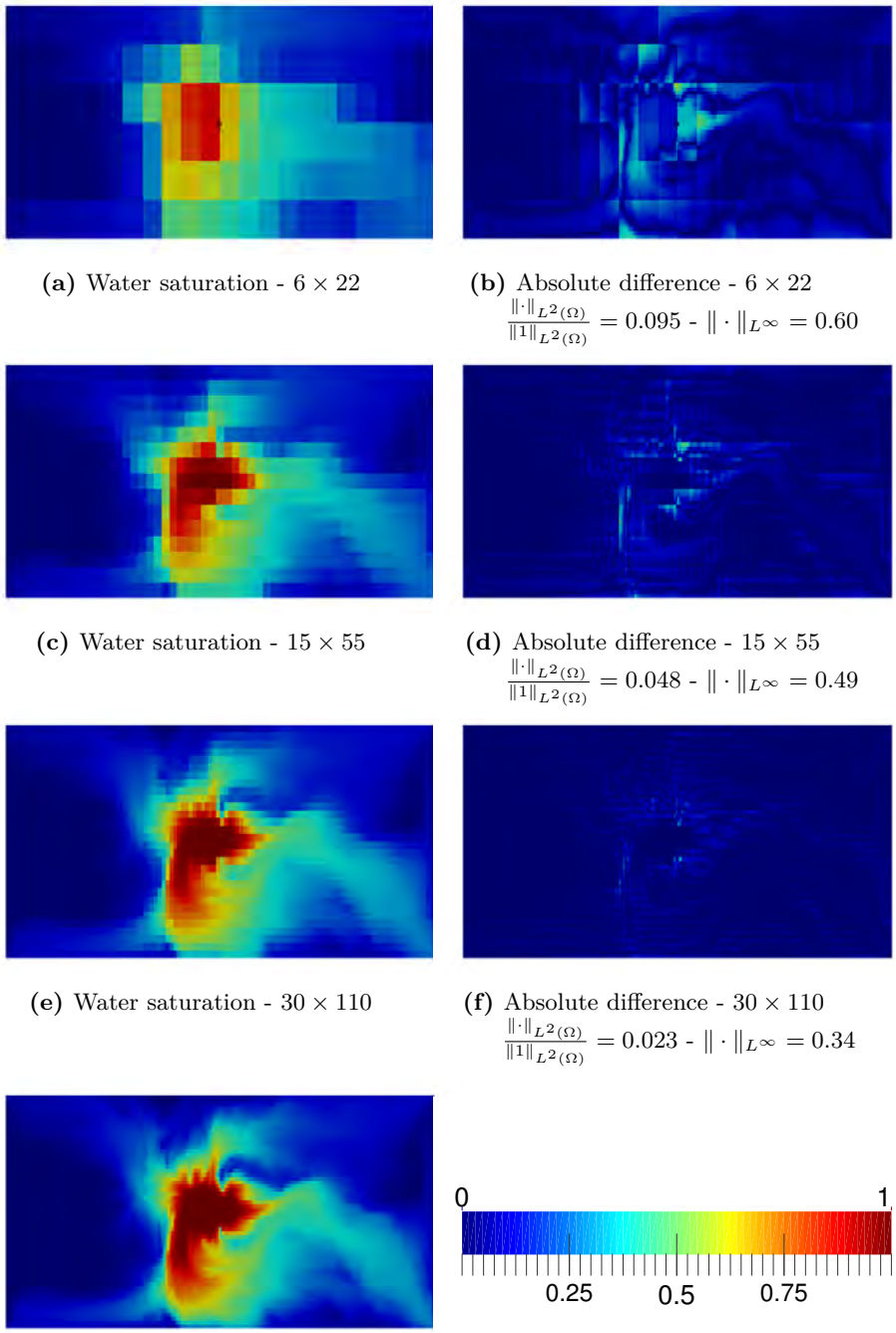
(d) Absolute difference - 15×55
$\frac{\|\cdot\|_{L^2(\Omega)}}{\|1\|_{L^2(\Omega)}} = 0.022$ - $\|\cdot\|_{L^\infty} = 0.42$

(e) Water saturation - 30×110

(f) Absolute difference - 30×110
$\frac{\|\cdot\|_{L^2(\Omega)}}{\|1\|_{L^2(\Omega)}} = 0.011$ - $\|\cdot\|_{L^\infty} = 0.22$

(g) Water saturation - 60×220

Figure 6.7: Left column: Water saturation after first water breakthrough (after 210 days) for fine grid reference solution and the 3 upscaled solutions. Right column: Absolute difference between fine grid reference solution and 3 upscaled solutions.



(a) Water saturation - 6×22

(b) Absolute difference - 6×22
$\frac{\|\cdot\|_{L^2(\Omega)}}{\|1\|_{L^2(\Omega)}} = 0.095$ - $\|\cdot\|_{L^\infty} = 0.60$

(c) Water saturation - 15×55

(d) Absolute difference - 15×55
$\frac{\|\cdot\|_{L^2(\Omega)}}{\|1\|_{L^2(\Omega)}} = 0.048$ - $\|\cdot\|_{L^\infty} = 0.49$

(e) Water saturation - 30×110

(f) Absolute difference - 30×110
$\frac{\|\cdot\|_{L^2(\Omega)}}{\|1\|_{L^2(\Omega)}} = 0.023$ - $\|\cdot\|_{L^\infty} = 0.34$

(g) Water saturation - 60×220

Figure 6.8: Left column: Water saturation after 10 years for fine grid reference solution and the 3 upscaled solutions. Right column: Absolute difference between fine grid reference solution and 3 upscaled solutions.

Level	#DoFs	#DoFs($\mathbf{u}$)	#DoFs(p)	nnz
0 (fine)	92680	53080	13200	607485
1	9107	6614	831	107355
2	2529	1896	211	37213
3	746	575	57	13346
4	233	179	18	4240

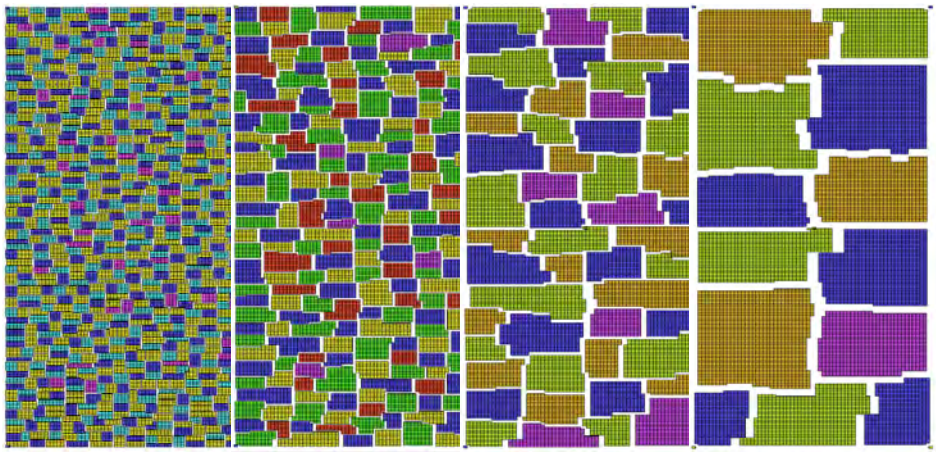
Table 6.4: Degrees of freedom and number of non-zeros for recursive upscaling of the top layer of SPE10.

ber of degrees of freedom for the upscaled problem relative to the number of degrees of freedom for the fine grid problem). The plots in Figure 6.10 show the $L^2(0,T;X)$ error norm (left) and $L^\infty(0,T;X)$ error norm (right), defined as

$$\begin{aligned} \|x_h - x_H\|_{L^2(0,T;X)} &= \sqrt{\int_0^T \|x_h(t) - x_H(t)\|_X^2 dt}, \\ \|x_h - x_H\|_{L^\infty(0,T;X)} &= \max_{t\in[0,T]} \|x_h(t) - x_H(t)\|_X. \end{aligned} \tag{6.37}$$

Here x_h and x_H are generic notation for the fine grid unknowns p_h, u_h, $(S_o)_h$, $(S_w)_h$ and the upscaled unknowns p_H, u_H, $(S_o)_H$, $(S_w)_H$, respectively. Similarly, X is a generic notation for the functional spaces $H(\mathrm{div},\Omega)$, $L^2(\Omega)$, $L^2(\Omega)$, $L^2(\Omega)$ were the unknowns are defined.

Both error norms are normalized by the appropriate norm of the fine grid reference solution. The exact formula is given by the labels in the figure. Figure 6.10 suggests that our upscaling approximates the solution very well: increasing errors as a function of coarsening is inevitable since we are necessarily losing information, but the sublinear rate of such increase demonstrates that the AMGe coarse spaces lead to more accurate results than solving the problem using standard finite elements on an equally coarse grid (i.e. for the same number of degree of freedom). Finally, the number of degrees of freedom and the number of non-zeros for the upscaled problems is given in Table 6.4. As evident from Table 6.4, the multilevel recursive upscaling provides a nice reduction in both the number of degrees of freedom and the number of non-zeros. In fact, the arithmetic complexity for the overall multilevel upscaling (all levels included) is 1.14 and similarly the operator complexity is 1.27.



(a) Level 1 **(b)** Level 2 **(c)** Level 3 **(d)** Level 4

Figure 6.9: Unstructured (METIS) recursive agglomeration of the SPE10 top layer. Elements containing wells are unagglomerated.

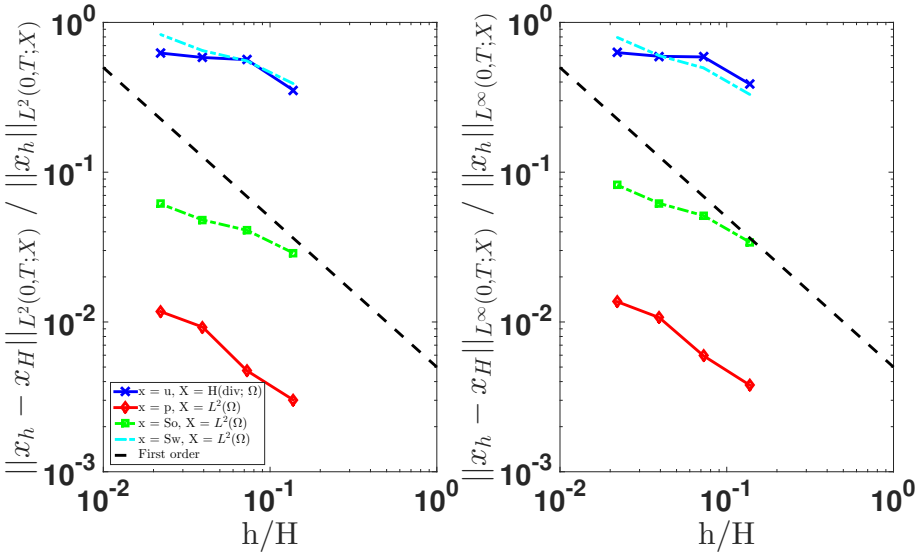


Figure 6.10: Error norms (6.37) as a function of the coarsening ratio h/H, defined as the square root of the ratio between the number of degrees of freedom for the upscaled system and the number of degrees of freedom for the fine grid system.

6.5.2 Modified SAIGUP

The SAIGUP study is a project with the purpose to *quantify objectively the sensitivity of geological complexity on production forecasts, as a function of generic aspects of both the sedimentological architecture and faulted structure of shallow marine hydrocarbon reservoirs* and *to validate these results using real-case reservoir and production data*, [44]. In this work, we use the same geometry and permeability field as in the SAIGUP benchmark in order to apply our upscaling techniques to a more realistic and geometrically challenging case than the SPE10.

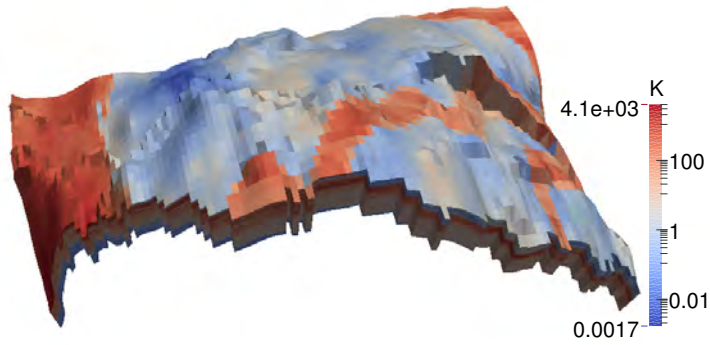
The SAIGUP mesh contains faults, which – at present – we are not able to account for. For this reason, the original SAIGUP is remeshed to remove the faults in the reservoir. This is done by extracting the (x, y) coordinates for all the vertices and the z coordinates for the vertices of top and bottom surfaces and then create the remaining z coordinates in between by interpolation so that the original number of layers in the mesh is obtained. This procedure results in the mesh illustrated in Figure 6.11a (plotted with the permeability field) and in Figure 6.11b (plotted with the wells). Note that, for visualization purposes, we rescaled the mesh to emphasize the vertical features of the geometry, but in reality it is much flatter.

The remeshing procedure results in cells with bad aspect ratio (~ 20), where the faults were originally located, however aspects ratios in reservoir simulation grids can be even worse. The rest of the cells have aspect ratios around 6. This could be remedied by cutting cells into two or more, but this has not been done for this work.

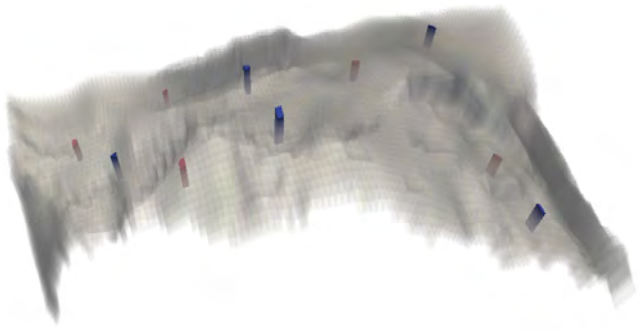
There are 5 production wells, which all are perforated in the top 14 layers. All production wells have a bottom hole pressure of 175 bar. There are 5 water injection wells, which all are perforated in the bottom 12 layers. All the perforations of the injection wells are set to inject 0.0058 times the element volume per day. This is a total injection of 7892 m^3 of water per day. The simulation horizon is 30 years with constant time step sizes of 30 days. This means after 30 years we have injected roughly 15% of the pore volume ($\phi \cdot V = 0.3 \cdot 1.85\,\mathrm{km}^3$) of the mesh. Since the main interest of the paper is on the numerical upscaling, we have purposely kept the time stepping very simple.

Unstructured (using METIS) and structured coarsening is applied to this mesh to form the agglomerates. Four different agglomeration strategies have been applied:

- **Full coarsening** - Graph partitioning algorithm is called directly on the



(a) Permeability field K_x



(b) Water injection wells are marked with red and production wells are marked with blue.

Figure 6.11: Modified SAIGUP permeability field and well locations (scaled with 0.5x in y-direction and 6x in z-direction).

Problem	#elements	#faces	#DoFs	nnz	arithmetic complexity	operator complexity
Fine grid	78720	243576	479736	3549946	-	-
Full coarsening (4)	17210	82081	168060	3039542	1.41412	1.85622
Full coarsening (16)	4920	25633	65422	1620930	1.17211	1.45661
Full coarsening (4)*	17960	84401	171960	3054520	1.4221	1.86044
Full coarsening (16)*	5616	28108	69924	1697050	1.18211	1.47805
Semi coarsening (4)*	14761	73982	173180	4080239	1.44573	2.14938
Semi coarsening (16)*	5629	30408	81513	2564226	1.21798	1.72233
Cartesian semi (4)*	20660	64372	175336	2194200	1.41582	1.61809
Cartesian semi (16)*	5970	19001	61025	901477	1.1523	1.25394
Cartesian semi (64)*	2100	6653	20683	318655	1.05114	1.08976

Table 6.5: Degrees of freedom, number of non-zeros and complexities. * means elements with wells and immediate neighbor elements of wells are left unagglomerated.

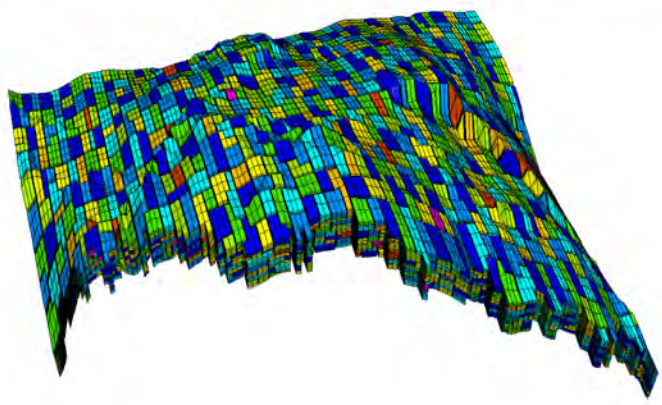
element to element connectivity graph without any preprocessing to take into account wells. (Figure 6.12a).

- **Full coarsening with wells** - Well elements and neighbor elements of wells are removed from the connectivity graph and left unagglomerated on the coarse mesh.
- (x, y)**-semi-coarsening with wells** - We remove inter-element connections in the graph if the normal of the shared face is almost vertical. In addition we do not agglomerate elements with wells and their neighbors (Figure 6.12b).
- **Structured** (x, y)**-semi-coarsening with wells** - Utilizing the underlying cartesian topology of the mesh, structured partitioning is used. This strategy is denoted "Cartesian semi" (Figure 6.12c).

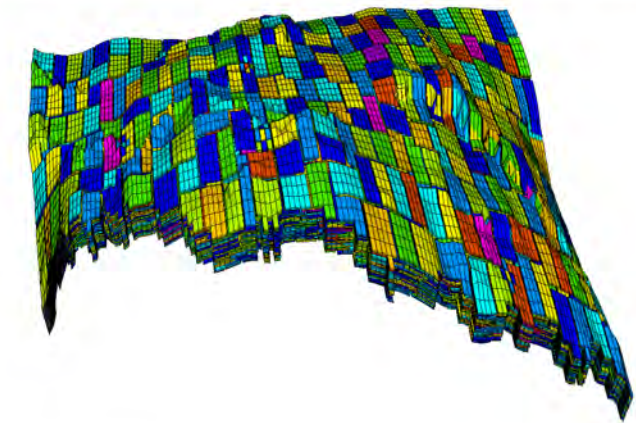
Note that we use a coloring algorithm to show the agglomerated elements. Table 6.5 contains information about the #DoFs, number of non-zeros and complexities for the different types of coarsening.

Figures 6.13a and 6.13b show the daily and accumulated production data for the fine grid reference solution and the three different agglomeration strategies. The number in parenthesis, (4), (16) and (64), indicates how many fine grid elements are grouped into one agglomerate.

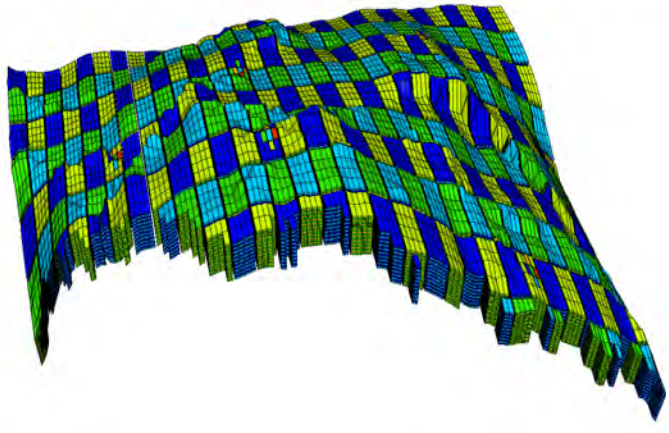
Figures 6.14a and 6.14b show the difference between the upscaled solutions and the fine grid reference solution. Given the appropriate coarsening strategy we are able to approximate the fine grid reference solution with a very good accuracy. The choice of agglomeration strategy clearly has a big impact on the accuracy of the upscaled solution. By leaving the elements and neighbor elements of the wells unagglomerated, the upscaled solutions are more accurate. Furthermore, due to the strong coupling in the vertical direction, applying (x,y)-semi-coarsening also provides an even more accurate upscaled solution. However, by looking at Table 6.5, we can see that the operator complexity, are



(a) Full coarsening

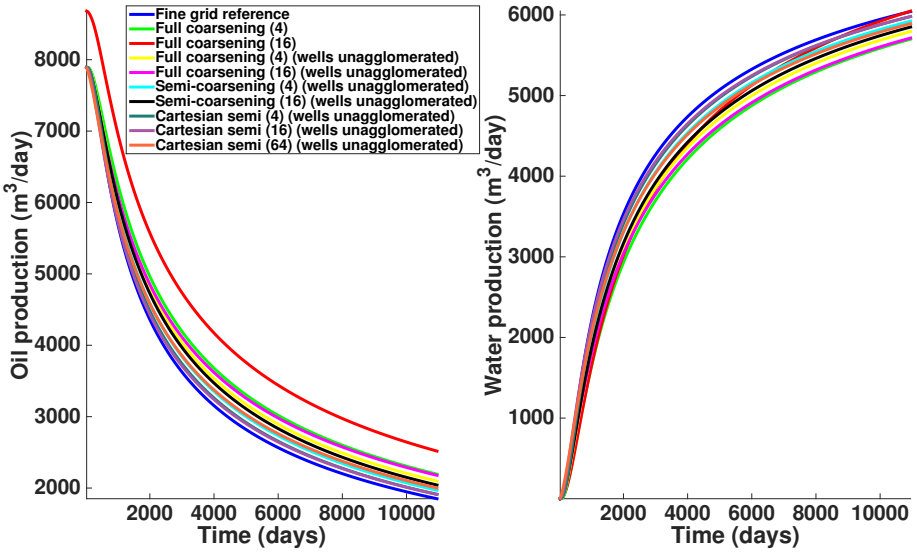


(b) (x, y)-semi-coarsening

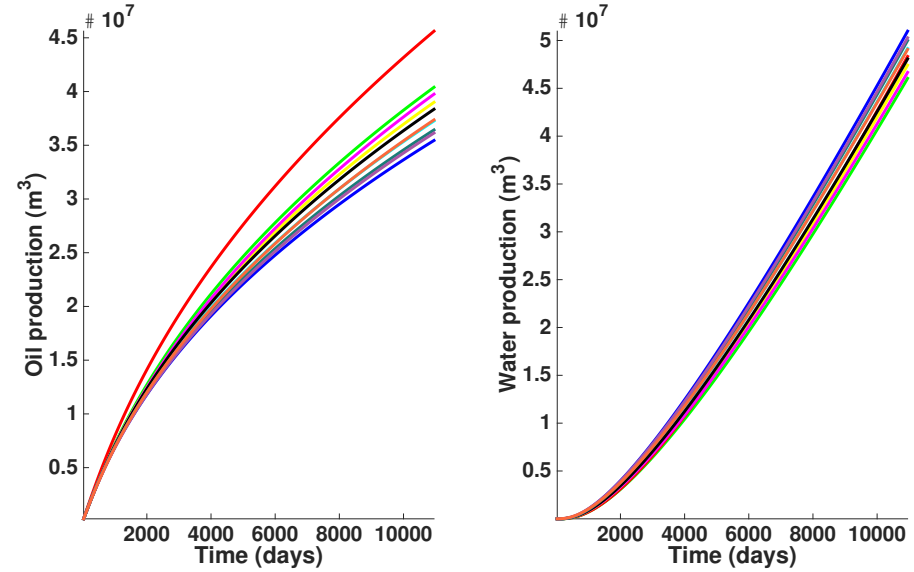


(c) Structured (x, y)-semi-coarsening (Cartesian semi)

Figure 6.12: Modified SAIGUP agglomerates (16 fine elements per agglomerate).



(a) Daily production data for fine grid and upscaled models.



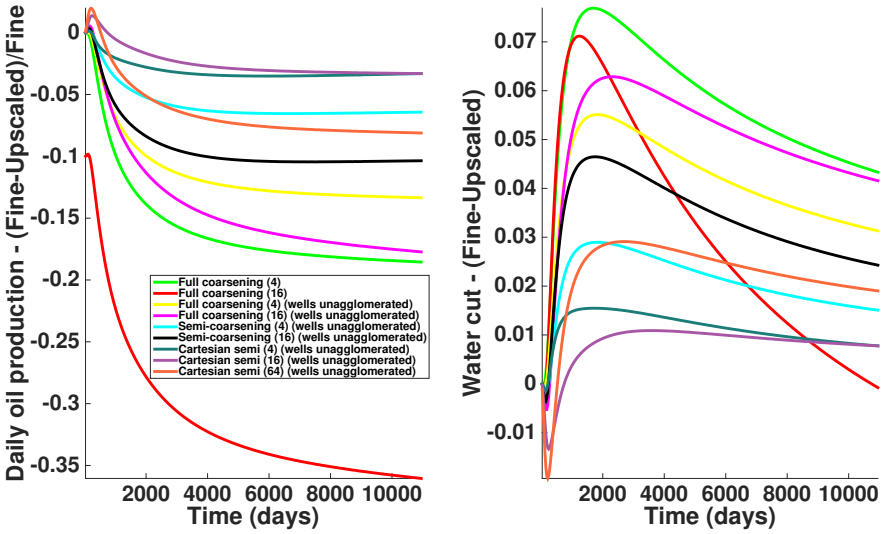
(b) Accumulated production data for fine grid and upscaled models.

Figure 6.13: Daily and accumulated production data for the modified SAIGUP model.

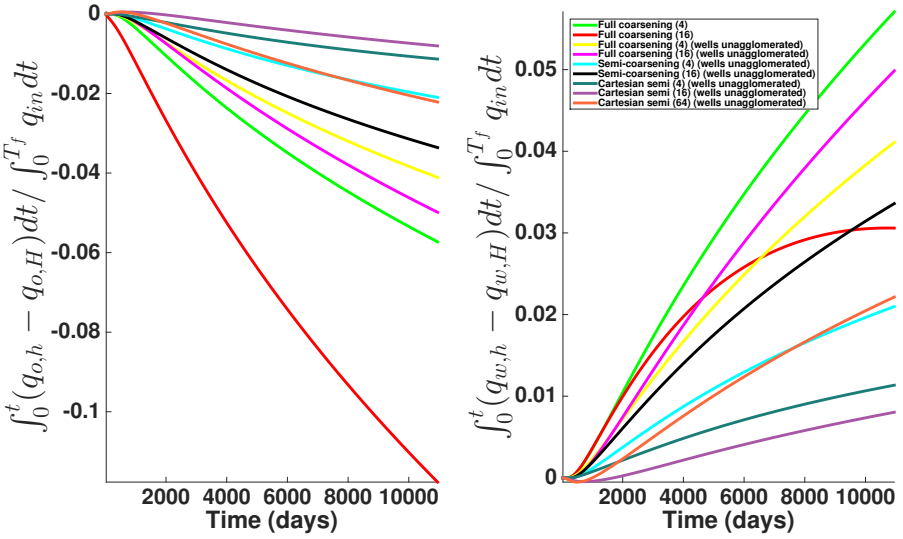
a bit high for the unstructured semi-coarsening. By utilizing the underlying cartesian topology of the mesh to generate a structured partitioning, the operator complexities drops significantly. This is due to two effects: on one hand, with cartesian agglomeration coarse faces are fewer and tend to be more planar, leading to a reduction in the number of degrees of freedom (as it can also be seen by the decrease in arithmetic complexity); on the other hand each agglomerated element tends to have less coarse faces resulting in smaller dense elemental matrices and therefore in a sparser global upscaled mixed system. This stresses the importance of a good quality of the agglomerated mesh. A general purpose software like METIS allows a lot of flexibility at the cost of a more expensive upscaled system. For this reason, we advocate whenever possible to exploit all the information of the fine grid topology and problem parameters to improve both accuracy and computational efficiency of the upscaled model.

In Figures 6.14a and 6.14b, it can be seen that the best compromise (in these tests) between complexities and accuracy is the cartesian semi-coarsening strategy with 64 fine elements per agglomerate. The maximum error in daily production is around 7% for oil and 2% for water with an operator complexity of only 1.09.

Figure 6.15 shows the error (compared to the fine grid solution) in water saturation after 30 years of injection for the upscaled solutions using the cartesian semicoarsening. As stated previously, the upscaling results in a more diffusive discretization, however the upscaled models still capture the solution well. For this particular problem and choice of coarsening strategy, it seems the error in the infinity norm is consistently around 0.87 probably due to a problematic area around the green well in the top right corner. Table 6.6 provides information on the accuracy of the upscaled results for the 5 individual production wells. Specifically, the table reports for each well the time of water breakthrough and the total production error computed as $\int_0^{T_f}(q_{o,h} - q_{o,H})dt / \int_0^{T_f} q_{o_{total},h}dt$, where the final time T_f is 30 years and $\int_0^{T_f} q_{o_{total},h}dt$ is the total amount of oil produced in all wells. It is evident that the tendency to predict earlier water breakthrough remains. Furthermore, the oil production errors range from 0.02% (well 5 with cartesian semicoarsening (4)) to 6.5% (well 3 with full coarsening (16)). These errors are highly dependent on the choice of agglomerates. It is worth noting that the 6.5% error is in the case where agglomeration does not take the wells into consideration. This highlights the importance of keeping well cells (and possibly their neighbor cells) unagglomerated.

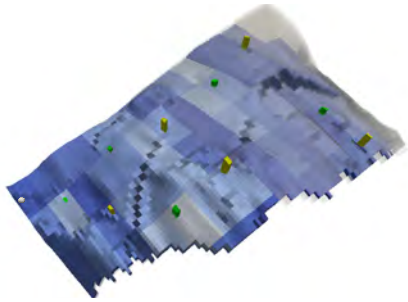


(a) Difference between fine grid and upscaled models in terms of daily production and water cut.

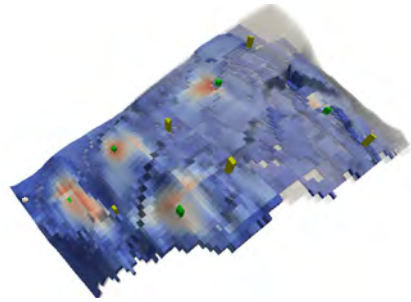


(b) Difference in production between the fine grid and upscaled models integrated over time relative to the total injection of water over time.

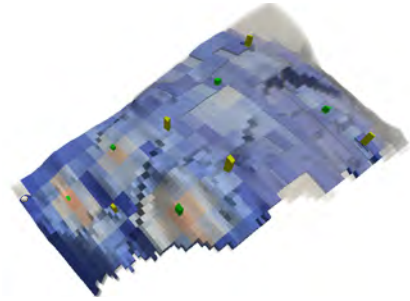
Figure 6.14: Difference between fine grid and upscaled productions for the modified SAIGUP model.



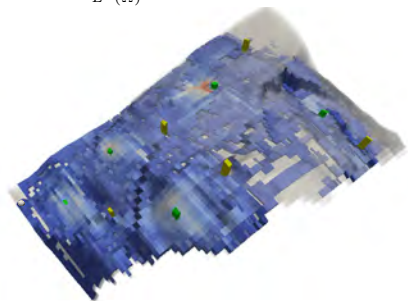
(a) Water saturation - CF: 64



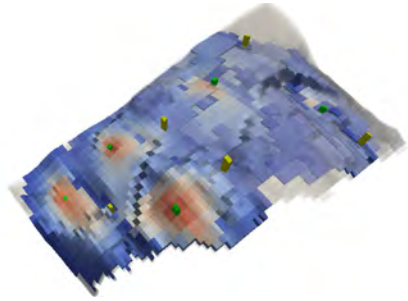
(b) Absolute difference - CF: 64
$\frac{\|\cdot\|_{L^2(\Omega)}}{\|1\|_{L^2(\Omega)}} = 0.101$ - $\|\cdot\|_{L^\infty} = 0.8788$



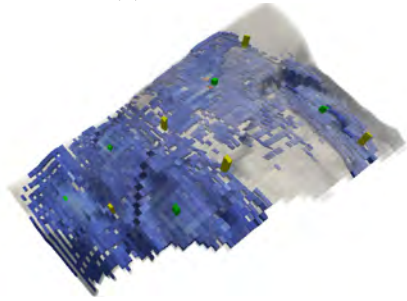
(c) Water saturation - CF: 16



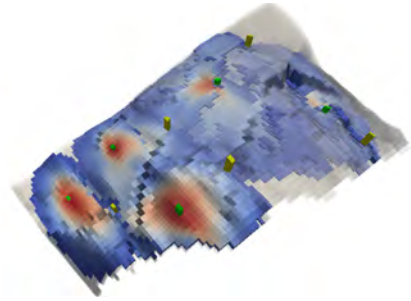
(d) Absolute difference - CF: 16
$\frac{\|\cdot\|_{L^2(\Omega)}}{\|1\|_{L^2(\Omega)}} = 0.061$ - $\|\cdot\|_{L^\infty} = 0.8703$



(e) Water saturation - CF: 4



(f) Absolute difference - CF: 4
$\frac{\|\cdot\|_{L^2(\Omega)}}{\|1\|_{L^2(\Omega)}} = 0.037$ - $\|\cdot\|_{L^\infty} = 0.8735$



(g) Water saturation - Fine grid

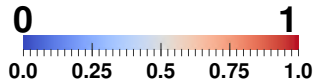


Figure 6.15: Left column: Water saturation after 30 years for fine grid reference solution and the 3 upscaled solutions. Right column: Abso-

Water breakthrough (days)	Well 1	Well 2	Well 3	Well 4	Well 5
Fine grid	180	480	450	120	630
Full coarsening (4)	210	420	450	120	660
Full coarsening (16)	240	390	390	90	540
Full coarsening (4)*	210	450	480	120	690
Full coarsening (16)*	180	360	390	90	540
Semi coarsening (4)*	180	420	420	90	420
Semi coarsening (16)*	180	360	330	90	330
Cartesian semi (4)*	180	420	390	90	480
Cartesian semi (16)*	150	390	330	60	330
Cartesian semi (64)*	210	300	300	60	180
Oil production error					
Full coarsening (4)	-0.0044	-0.0385	-0.0341	-0.0041	-0.0160
Full coarsening (16)	-0.0597	-0.0458	-0.0649	-0.0160	-0.0130
Full coarsening (4)*	-0.0174	-0.0207	-0.0157	-0.0065	-0.0094
Full coarsening (16)*	-0.0018	-0.0412	-0.0254	-0.0102	-0.0059
Semi coarsening (4)*	-0.0198	-0.0048	-0.0050	-0.0039	-0.0021
Semi coarsening (16)*	-0.0388	-0.0077	-0.0071	-0.0038	0.0005
Cartesian semi (4)*	-0.0129	-0.0018	-0.0026	-0.0017	-0.0002
Cartesian semi (16)*	-0.0212	-0.0018	-0.0020	0.0044	0.0070
Cartesian semi (64)*	-0.0548	-0.0067	-0.0056	0.0119	0.0177

Table 6.6: Water breakthrough is defined as the time, where 1% of the produced fluids is water. The oil production error is computed based on the accumulated production per well: $\int_0^{T_f}(q_{o,h} - q_{o,H})dt / \int_0^{T_f} q_{o_{total},h}dt$. * means elements with wells and immediate neighbor elements of wells are left unagglomerated.

6.5.2.1 Multilevel recursive upscaling

In this section, the multilevel recursive upscaling capability of the AMGe method is demonstrated for the modified SAIGUP case. The premise for this study is the same as described in Section 6.5.1.1. METIS is used to form the agglomerated meshes. Five levels including the fine grid are used. A coarsening factor of sixteen is used from level zero to level one and a coarsening factor of four is used for the remaining agglomeration steps.

The error norms for the difference between the upscaled solutions and the fine grid reference solution normalized by the norm of the fine grid reference solution is plotted in Figure 6.16. On the x-axis of the figure we report the agglomeration factor h/H roughly estimated as the cubic root of the ratio between the number of degree of freedom of the upscaled problem and fine grid problem. An alternative, which is outside the scope of this paper, is to study the accuracy of the coarse solution as a function of the computational cost (time-to-solution) for fine and upscaled model (see our results in [77]).

Figure 6.16 suggests that the upscaled velocity and the saturations converges linearly to the fine grid reference solution with respect to the coarsening ratio. On the other hand, we observe larger errors for the upscaled pressure for high coarsening factors. This deterioration in the approximation properties of the pressure space may be alleviated by enriching the pressure coarse space with additional coarse degree of freedom.

The number of degrees of freedom and complexities of the upscaled problem are given in Table 6.7. Considering all levels, the overall arithmetic complexity is 1.22 and the overall operator complexity is 1.80. Although the operator complexity is quite larger compared to the SPE10 case, this result is satisfactory considering the complexity of the 3D geometry.

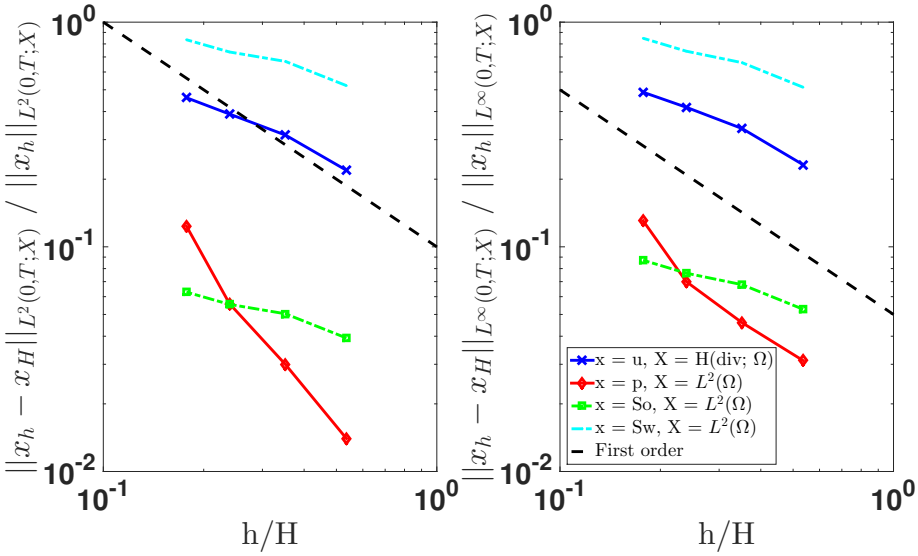


Figure 6.16: Error norms (6.37) as a function of the coarsening ratio h/H, defined as the cubic root of the ratio between the number of degrees of freedom for the upscaled system and the number of degrees of freedom for the fine grid system.

Level	#DoFs	#DoFs(**u**)	#DoFs(p)	nnz
0 (fine)	479736	243576	78720	3549946
1	73806	58638	5056	1864292
2	20865	16881	1328	648277
3	6550	5242	436	235884
4	2683	2023	220	102223

Table 6.7: Degrees of freedom and number of non-zeros for recursive upscaling of modified SAIGUP.

6.6 Summary

A version of the Element-based Algebraic Multigrid (AMGe) with guaranteed approximation properties for the coarse velocity spaces has successfully been applied to upscale a mixed formulation of the incompressible reservoir simulation equations. The method has demonstrated the ability to accurately approximate

the solution using significantly fewer degrees of freedom than that of the original system. More importantly, the nonzero entries of the resulting coarse (upscaled) problem, measured by the operator complexity, stays much less than two. This means that the memory requirement for storing the upscaled problem is (much) less than the storage needed for the fine-grid one. Two challenging test cases have been used to demonstrate this. Multilevel results show that the errors as a function of coarsening increase at a sublinear rate and therefore demonstrate that the AMGe coarse spaces lead to more accurate results than solving the same problem using standard finite elements on an equally coarse grid (i.e. for the same number of degree of freedom). One of the important challenges is to accurately predict when the water reaches a production well. The upscaled simulations compute oil and water production curves that accurately approximate the ones computed on the fine grid, however they tend to underestimate the exact time when water breakthrough happens due to the higher numerical diffusivity of the upscaled discretization. The experiments we performed have shown that the agglomeration strategy (grouping of the fine grid elements into agglomerates to get a coarse mesh) has a large impact on the resulting upscaled approximation. It is important to leave the elements containing wells and (possibly) their immediate neighbors unagglomerated to capture the near-well flow accurately. Furthermore, due to the strong coupling in the vertical direction, only agglomerating in the x- and y-directions gives a significantly better upscaled approximation. Finally, we have demonstrated that the method can be used for multilevel upscaling to generate a hierarchy of coarser models.

6.7 Perspectives

In this study we have mainly focused on the accuracy of the proposed upscaling method to demonstrate the applicability of the improved AMGe method for reservoir simulation. The computational efficiency of the method is the subject of a follow-up paper, where all time-to-solution aspects, which are both hardware and implementation dependent will be studied in depth. The computational benefits of the presented approach will be best utilized in a parallel computing setting. Improved parallelization of the software with MPI-OpenMP is an ongoing activity. These techniques are particularly well suited for modern multicore architectures, because the construction of the coarse spaces by solving many small local problems offers a high level of concurrency in the computations. Higher-order upscaling methods would further improve computational efficiency, because higher-order provides more computationally intensive local operations, meaning the relative communication overhead is not as significant as it is for low-order.

The solution of the coarse-grid mixed systems is also an important issue. We stress upon the fact that for very large-scale problems, even the coarse grid problem may still be fairly large and therefore direct solvers will not be a feasible approach. The coarse grid mixed systems can in fact be more ill-conditioned than the original fine grid system, which means solvers which work fine for the fine grid problem (such as the block diagonal AMG preconditioner) may experience difficulties with the coarse grid systems. This is due to the possibly complicated geometry of the agglomerates (and possibly, the uneven distribution of coarse DoFs within each agglomerate). We have ongoing activities in applying AMGe techniques to develop more robust preconditioners for the upscaled systems. In this way, the coarse spaces used for upscaling can be reused for preconditioning purposes.

Another possible direction is to construct adaptive coarsening strategies. As we have already observed in Section 6.4.4 our AMGe approach allows to locally increase the spatial resolution of the upscaled solution by either locally enriching the coarse degrees of freedom in a particular agglomerated element (similar to "p"-refinement for finite elements) or to use smaller agglomerates (similar to "h"-refinement). In addition, effective a posterior error estimators can be efficiently computed by exploiting the multilevel nature of our method.

Last but not least, future research directions include extending our framework to fully-coupled (pressure-velocity-saturations) implicit in time integrators. This will include using our multilevel hierarchies of coarse spaces with guaranteed approximation properties needed to construct efficient non-linear multigrid solvers (such as Full Approximation Scheme).

Paper III - Multilevel Techniques Lead to Accurate Numerical Upscaling and Scalable Robust Solvers for Reservoir Simulation

---

Authors: Max la Cour Christensen, Umberto Villa and Panayot Vassilevski.

Presented at SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA, 23-25 February, 2015.

Chapter 7

Paper III - Multilevel Techniques Lead to Accurate Numerical Upscaling and Scalable Robust Solvers for Reservoir Simulation

Abstract: This paper demonstrates an application of element-based Algebraic Multigrid (AMGe) technique developed at LLNL ([115]) to the numerical upscaling and preconditioning of subsurface porous media flow problems. The upscaling results presented here are further extension of our recent work in [33]. The AMGe approach is well suited for the solution of large problems coming from finite element discretizations of systems of partial differential equations. The AMGe technique from [73, 72] allows for the construction of operator-dependent coarse (upscaled) models and guarantees approximation properties of the coarse velocity spaces by introducing additional degrees of freedom associated with non-planar interfaces between agglomerates. This leads to coarse spaces which maintain the specific desirable properties of the original pair of Raviart-Thomas and piecewise discontinuous polynomial spaces. These coarse spaces can be used both as an upscaling tool and as a robust and scalable

solver. The methods employed in the present paper have provable O(N) scaling and are particularly well suited for modern multicore architectures, because the construction of the coarse spaces by solving many small local problems offers a high level of concurrency in the computations. Numerical experiments demonstrate the accuracy of using AMGe as an upscaling tool and comparisons are made to more traditional flow-based upscaling techniques. The efficient solution of both the original and upscaled problem is also addressed, and a specialized AMGe preconditioner for saddle point problems is compared to state-of-the-art algebraic multigrid block preconditioners. In particular, we show that for the algebraically upscaled systems, our AMGe preconditioner outperforms traditional solvers. Lastly, parallel strong scaling of a distributed memory implementation of the reservoir simulator is demonstrated.

7.1 Introduction

Numerical simulation of subsurface flow problems features a multi-physics and multi-scale nature that poses a substantial challenge to state-of-the-art solvers. In addition, in many applications of practical interest the physical properties of the medium are not known a priori and may feature a stochastic nature. To obtain reliable results, numerical methods must address such sources of uncertainty often requiring repeated solutions for different realization of the unknowns parameters. Upscaling techniques can reduce computational cost by solving coarse scale models that take into account interactions at different scales. Classical upscaling techniques relies on averaging or homogenization techniques for the coefficient of the partial differential equation. However, only limited attempts have been made to analyze those techniques, providing almost no indication of the quality of the approximation or no verification whether the assumptions made in deriving the upscaled coefficient hold[34]. On the other hand, recent advances in multiscale methods and variational upscaling[58, 121, 97, 91] allows for more reliable and accurate results by projecting the fine grid problem on appropriate lower dimensional subsets of the original space.

In this work, we apply an element-agglomeration algebraic multigrid (or AMGe for short) framework to coarsen a wide class of partial differential equations (such as elliptic equations, transport equations, Darcy equations, and Maxwell equations) on general unstructured meshes using mixed finite element setting. Such AMGe framework was first introduced in [99] for preconditioning purposes and it allows for the construction of coarse spaces which are operator-dependent. The method was then extended in [73, 72] in order to guarantee approximation properties of the coarse AMGe spaces. The main advantage of this framework is that it is by its very nature multilevel and that it can be applied for both

upscaling as well as for the construction of robust multilevel linear and nonlinear solvers.

In addition, the great flexibility of the AMGe framework enables fully upscaled systems, which we believe is essential for efficient solution of very large-scale models. In fact, often only the pressure and total velocity unknowns are upscaled, but the saturation equation is still solved on a fine grid level[58, 121, 97, 91]. On the contrary, our approach allows upscaling to be applied also to the saturation equations, using the same coarse spaces computed for upscaling the pressure unknown. This ability to simulate, in as easy and flexible manner, at different spatial resolutions (with optimal computational costs) highly benefits applications to uncertainty quantification (e.g. based on Multilevel Monte Carlo methods) and optimization, which require to repeatedly solve linear systems for different realizations of the parameters.

In what follows, we first describe the governing equations and the discretization we employ. We continue next with a brief introduction to the choice of numerical methods and to a state-of-the-art preconditioner (referred to as $L_2 - H^1$ block preconditioner) for the saddle-point systems targeted in this paper. After this, the specific type of AMGe used in this work is described. The last part of the paper contains the numerical experiments and their analysis. Specifically, the employed AMGe is used both as an upscaling tool and as a preconditioner. To show the superior accuracy of AMGe as an upscaling tool, it is compared to a more traditional upscaling. A comparison is also made between AMGe and the $L_2 - H^1$ block preconditioner to evaluate the algorithmic efficiency of the methods. More importantly, we demonstrate how AMGe also can be used to precondition the algebraically upscaled systems. This is of practical interest since for large-scale simulations, the upscaled systems may still be fairly large. We show that the solver using the $L_2 - H^1$-preconditioner deteriorates for the upscaled systems, whereas the AMGe maintains its efficiency. In the end, we present first results of our parallel code in progress, demonstrating strong scaling of the reservoir simulator. We conclude with some remarks.

7.2 Governing equations

Our model of interest is based on a total velocity formulation of two-phases incompressible flow in porous media[30]. By letting the total velocity $\mathbf{u}$, the pressure p and the saturations S_α ($\alpha = o$ for the oil phase and $\alpha = w$ for the water phase) be the primary variables of the problem, the system of governing

equations reads

$$\mathbf{K}^{-1}\lambda^{-1}(S)\mathbf{u} + \nabla p = \left(\sum_{\alpha} \rho_\alpha f_\alpha(S)\right)\mathbf{g} \tag{7.1}$$

$$\nabla \cdot \mathbf{u} - q(p, S) = 0 \tag{7.2}$$

$$\phi \frac{\partial S_\alpha}{\partial t} + \nabla \cdot \mathbf{u}_\alpha(S_\alpha) = \frac{q_\alpha}{\rho_\alpha}. \tag{7.3}$$

Here $\nabla\cdot$ is the divergence operator, $\mathbf{K}$ is the absolute permeability tensor, ϕ is the porosity, λ is the total mobility, $\mathbf{g}$ is the gravitational acceleration, and q is the total source term. $\mathbf{u}_\alpha$ is the phase velocity, f_α is the fractional flow function, ρ_α is the mass density, q_α is the source term for each phase α. We refer to [33] for the full description of the set of equations used in this work.

7.3 Discretization

The system of governing equations is discretized using a Mixed Finite Element method for the total velocity and the pressure equations (9.5)-(9.6) and a Discontinuous Galerkin method for the saturation equations (9.7).

We use standard notation. For scalar functions $p, w \in L^2(\Omega)$ and vector functions $\mathbf{u}, \mathbf{v} \in \mathbf{L}^2(\Omega) = [L^2(\Omega)]^d$, we define the inner products

$$(p, w) = \int_\Omega p\, w\, d\Omega \quad \text{and} \quad (\mathbf{u}, \mathbf{v}) = \int_\Omega \mathbf{u} \cdot \mathbf{v}\, d\Omega.$$

Furthermore, we define the functional spaces $\mathbf{R}$ and W as

$$\mathbf{R} \equiv H(\mathrm{div}; \Omega) := \{\mathbf{u} \in L^2(\Omega) \,|\, \mathrm{div}\, \mathbf{u} \in L^2(\Omega) \text{ and } \mathbf{u} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega\},$$
$$W \equiv L^2(\Omega).$$

By letting $\mathbf{R}_h \subset \mathbf{R}$ denote the lowest order Raviart–Thomas finite element space and $W_h \subset W$ denote the finite element space of the piecewise constant functions[47], the finite element discretization of the governing equations (9.5) and (9.6) results in the following mixed variational problem.

PROBLEM 9 *Find* $(\mathbf{u}_h, p_h) \in \mathbf{R}_h \times W_h$ *such that*

$$\begin{cases} \left(\mathbf{K}_h^{-1}\lambda_h^{-1}\mathbf{u}_h, \mathbf{v}_h\right) - \left(p_h, \nabla\cdot\mathbf{v}_h\right) = \left(\mathbf{g}\sum_\alpha f_{\alpha,h}\rho_{\alpha,h}, \mathbf{v}_h\right), & \forall\ \mathbf{v}_h \in \mathbf{R}_h \\ \left(\nabla\cdot\mathbf{u}_h, w_h\right) - \left(q_h(p), w_h\right) = 0, & \forall\ w_h \in W_h \end{cases}$$

Problem 9 can be written, in algebraic form, as the following indefinite saddle point problem

$$\mathbf{A}\mathbf{X} = \mathbf{B}, \tag{7.4}$$

where the block matrix $\mathbf{A}$ and block vectors $\mathbf{X}$ and $\mathbf{B}$ read:

$$\mathbf{A} = \begin{bmatrix} M & B^T \\ B & -C \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{F_u} \\ F_p \end{bmatrix}. \tag{7.5}$$

Here $\mathbf{u}$ and p collect the degree of freedom of the finite element unknowns $\mathbf{u}_h$ and p_h, M is the weighted velocity mass matrix, B stems for the discretization of the divergence operator, and C is a semi-positive diagonal matrix whose nonzeros entries are used to model wells governed in bottom hole pressure. Efficient methods for the solution of the saddle point system (7.4) will be discussed later in the paper.

Concerning the discretization of the saturation equations (9.7), we use the forward Euler method in time and the Discontinuous Galerkin Finite Element Method in space. For some given total velocity $\mathbf{u_h}^*$ and pressure p_h^*, the discretized saturation equation for phase α reads

PROBLEM 10 *Find* $S_h^{n+1} \in W_h$ *such that*

$$\frac{1}{\Delta t}\left(\phi S_{\alpha,h}^{n+1}, w_h\right) = \frac{1}{\Delta t}\left(\phi S_{\alpha,h}^{n}, w_h\right) + F_\alpha(S_h^n, \mathbf{u_h}^*, w_h) + \left(\frac{q_\alpha(S_h^n, p_h^*)}{\rho_\alpha}, w_h\right) \quad \forall\ w_h \in W_h.$$

Here the flux term $F_\alpha(S, \mathbf{u}, w_h)$ is given by

$$F_\alpha(S, \mathbf{u}, w_h) = \sum_{f\in\mathcal{F}_h} \int_f (\mathbf{u}_\alpha(S_\alpha, \mathbf{u})\cdot\mathbf{n})^{up}\ [w_h]\ dS, \tag{7.6}$$

where $\mathcal{F}_h$ is the set of all the faces in the mesh, $[w_h] = (w_h|_{e^-}) - (w_h|_{e^+})$ represents the jump of the test function across the face f shared by the elements e^- and e^+, and $(\mathbf{u}_\alpha(S_\alpha, \mathbf{u})\cdot\mathbf{n})^{up}$ stems from the upwind numerical flux computed using the approach described in [88]. We refer to [33] for further details.

Problem 10 can be written in algebraic form as

$$\frac{\phi}{\Delta t} W S_\alpha^{n+1} = \frac{\phi}{\Delta t} W S_\alpha^n + F_S + \frac{1}{\rho_\alpha} W q_\alpha, \tag{7.7}$$

where W denotes the mass matrix in the finite element space of piecewise constant functions W_h, and F_S stems for the discretization of the flux term in (7.6). It is worth noticing that the saturation update in (7.7) is fully explicit since, in our settings, the matrix W is diagonal.

Finally, the Improved IMPES[29] (IMplicit Pressure Explicit Saturations) method is used to weakly couple Problem 9 and Problem 10, so that $\mathbf{u_h}^*$ and p_h^* in Problem 10 denote the *frozen* total velocity and pressure obtained by solving Problem 9 at selected time steps.

7.4 Element-based Algebraic Multigrid (AMGe)

AMGe is a framework of multigrid methods for systems coming from finite element discretizations. The components in AMGe are constructed from local element information, such as finite element matrices and element topology. This is in contrast to algebraic multigrid (AMG), where only system coefficients are used to construct the hierarchy of coarse spaces.

In this section, we briefly summarize the AMGe technique for the construction of operator-dependent coarse spaces with guaranteed approximation properties on general unstructured grids, referring to [99, 73, 72] for the details.

7.4.1 The construction of the agglomerated meshes

Agglomerates are formed by grouping together fine-grid elements. To this aim, we build the dual graph of the mesh, which is an undirected graph, where each node of the graph represents an element in the mesh and node i is connected to node j if element i and element j share a face. METIS, [66], is used for the partitioning of the undirected graph resulting in agglomerates consisting of fine-grid elements. The coarse faces consists of the fine faces belonging to the

intersection of any pair of neighboring agglomerates. This means that for unstructured meshes the coarse faces are non-planar in most cases. This procedure applies to general unstructured meshes, as long as the resulting agglomerated elements and faces are post-processed to meet certain topological constraints.

7.4.2 The construction the coarse pressure and saturation spaces

The coarse pressure space $W_H \subset W_h$ consists of functions which are constant on each agglomerated element. In addition, one can enrich the space W_H by restricting additional functions that contain information regarding the specific problem. For example, one can add an additional function which has support on the elements containing a well or their neighbours. A basis of the space W_H is then used to form the columns of the prolongation matrix $\mathbf{P}_p : W_H \to W_h$. In the following, we assume that the saturations are also upscaled using the same coarse space W_H. This is consistent with the choice of finite element spaces for the fine grid discretization, but it is not necessary.

7.4.3 The construction of the coarse velocity space

The construction of the coarse velocity space $\mathbf{R}_H \subset \mathbf{R}_h$ is performed in two steps: first we prescribe the traces of the coarse velocity space on each agglomerate face F, and then we extend such traces to the interior of the neighboring agglomerated element subject to the constraint $\nabla \cdot \mathbf{R}_H = W_H$. This property is necessary to preserve the stability of the upscaled discretization and to guarantee that the spaces $(\mathbf{R}_H, W_H)$ are inf-sup compatible. To enforce approximation properties of the coarse velocity space, we provide the set of finite element functions $\{\mathbf{r_i}\}_{\mathbf{i=0}}^{\mathbf{n}}$ ($\mathbf{r_i} \in \mathbf{R_h}$) that the coarse space $\mathbf{R}_H$ needs to interpolate exactly. For example, to preserve the same approximation properties of the lowest order Raviart-Thomas finite element space, one should choose the three constant vectors $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ as the set of targets. For each agglomerated face F we also define the additional target $\mathbf{1}$, [99], such that $\mathbf{1} \cdot \mathbf{n}_f = \pm 1$, where $\mathbf{n}_f$ denotes the normal vector to the fine face f and the sign depends on the reciprocal orientation of the fine face f with respect to the agglomerated face F. In the first step of the construction, we restrict the target set $\{\mathbf{r}_i\}_{i=0}^{n} \bigcup \mathbf{1}$ on each agglomerated face F and we extract a linearly independent set $\{\boldsymbol{\phi}_i^H\}_{i=0}^{n'_F}$. In the second step of the construction, we loop over all the agglomerated elements T and, for each agglomerated face $F \subset \partial T$, we extend the local basis $\{\boldsymbol{\phi}_i^H\}_{i=0}^{n'_F}$ to the interior of

T by solving the local mixed system

$$\begin{cases} \left(\alpha\, \boldsymbol{\phi}_i^H, \mathbf{v}_h\right)_T + (p_h, \nabla \cdot \mathbf{v}_h)_T = 0, & \forall \mathbf{v}_h \in \mathbf{R}_h, \operatorname{supp} \mathbf{v}_h = T, \mathbf{v}_h \cdot \mathbf{n} = 0 \text{ on } \partial T \\ \left(\nabla \cdot \boldsymbol{\phi}_i^H, w_h\right)_T = 0, & \forall w_h \in W_h, \operatorname{supp} w_h = T, (w_h, 1) = 0. \end{cases}$$

Here the symmetric positive definite coefficient matrix α can be set equal to the coefficient of the original problem, $\mathbf{K}^{-1}\lambda^{-1}$, but this is not strictly necessary. Finally, we let the columns of the prolongation matrix $\mathbf{P_u} : \mathbf{R}_H \to \mathbf{R}_h$ be the collection of the coarse basis functions $\boldsymbol{\phi}_i^H$.

7.5 Preconditioning of the mixed system

In this section we describe two different approaches for the iterative solution of the discrete saddle point problem in (7.5). One is to use a symmetric positive definite block diagonal preconditioner and state-of-the-art algebraic multigrid solvers for each block; the other is a direct AMGe approach which exploits the hierarchy of our coarse spaces and respective Hiptmair smoothers.

7.5.1 The $L_2 - H^1$ preconditioner

An optimal block diagonal preconditioner for the saddle point system in (7.5) can be derived as follows. For any symmetric positive definite matrix H, we let $\widetilde{\Sigma} = BH^{-1}B^T + C$ and we define

$$\widetilde{\mathbf{A}} = \begin{bmatrix} H & B^T \\ B & -C \end{bmatrix}, \quad \text{and} \quad \mathbf{P} = \begin{bmatrix} H & 0 \\ 0 & \widetilde{\Sigma} \end{bmatrix}. \tag{7.8}$$

It is well-known that $\mathbf{P}$ is an optimal preconditioner for $\widetilde{\mathbf{A}}$, indeed in exact arithmetic preconditioned MINRES converges in 3 iterations[90]. By letting $H = \operatorname{diag}(M)$ and exploiting the fact that the finite element Raviart-Thomas mass matrix M is spectrally equivalent to its diagonal, it is possible to show that $\mathbf{P}$ is optimal also for the original problem $\mathbf{A}$. Moreover, since H is diagonal, the Schur Complement $\widetilde{\Sigma}$ is explicitly available and sparse, which allows us to approximate its inverse by a well-suited algebraic multigrid (AMG) solver. Specifically, we use BoomerAMG from hypre[53].

7.5.2 The AMGe Preconditioner

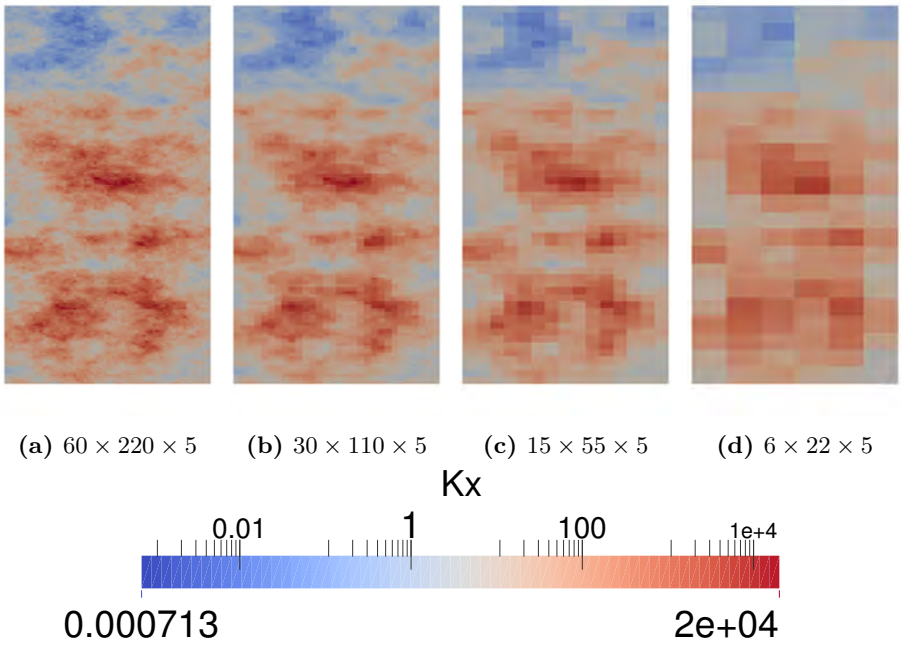
To construct an AMGe preconditioner specific for saddle point problems of the form of (7.5), we build a hierarchy of coarse velocity spaces $\mathbf{R}_k$ and respective pressure spaces counterparts W_k by recursively applying the procedure described in the previous section. The respective block matrices are labeled as $\mathbf{A}^{(k)}$ with index k corresponding to the finest level and $k = l$ to the coarsest level. In addition, we denote with $\mathcal{C}^{(k)}$ the discretization of the curl operator at level k (also provided by the AMGe coarsening), whose columns span the kernel of the divergence operator $B^{(k)}$. In the setup phase, for each subdomain T (which is a union of current level elements in the agglomerated mesh $\mathcal{T}_k$ at level k), we assemble the local problems $\mathbf{A}_T^{(k)}$ by imposing no flow boundary conditions on ∂T and we store their factorizations. Then the AMGe preconditioner consists in a symmetric V-cycle equipped with a sophisticated multiplicative smoother. More specifically, the smoothing procedure at level k consists of the following phases: (1) parallel local subdomain solves involving the non-overlapping subdomain matrices $\mathbf{A}_T^{(k)}$; (2) a divergence free correction of the velocity unknown at level k involving the symmetric semidefinite auxiliary matrix $(\mathcal{C}^{(k)})^T M^{(k)} \mathcal{C}^{(k)}$. To obtain a symmetric V-cycle we reverse the order of the two phases in the post-smoothing of the coarse grid correction.

7.6 Upscaling with AMGe

In this section, we present numerical results on a model consisting of the top 5 layers of the second dataset in the Tenth SPE Benchmark[95]. Specifically the permeability field in the x-direction is used as an isotropic permeability field. We compare the solution obtained by AMGe upscaling to the solution obtained by a flow-based upscaling method.

The software developed in this work uses the parallel C++ finite element library MFEM, [1], from Lawrence Livermore National Laboratory (LLNL). It supports a wide variety of finite element spaces in 2D and 3D, as well as many bilinear and linear forms defined on them. It includes classes for dealing with various types of triangular, quadrilateral, tetrahedral and hexahedral meshes and their global and local refinement. Parallelization in MFEM is based on MPI, and it leads to highly scalable finite element assembly procedures. It supports several solvers from the hypre library, [53], including the state-of-the-art algebraic multigrid preconditioner BoomerAMG.

Fig. 7.1 shows a top view of the fine grid SPE10 x-permeability field and of 3 up-



(a) $60 \times 220 \times 5$ **(b)** $30 \times 110 \times 5$ **(c)** $15 \times 55 \times 5$ **(d)** $6 \times 22 \times 5$

Figure 7.1: SPE10 top layer upscaled permeability fields using flow-based upscaling methods and fine grid permeability field. Note that for the upscaled permeability fields, only the x-component is shown here.

scaled permeability fields computed using a standard commercial software[105] for different level of coarsening. The software applies a flow-based method to generate an anisotropic upscaled permeability field on the coarse mesh. Note that only the x-component of the upscaled permeability fields is shown in Fig. 7.1.

For the simulations, we assume a constant porosity of 0.3 and an initial saturation of oil equal to one in all the domain. We consider a classic five wells pattern, in which four production wells are placed in each corner of the xy-plane and one water injection well in the center. The production wells are controlled by a bottom hole pressure of 275 bar and the injection well has a constant injection rate of 0.2 times the element volume. All wells have a radius of 0.2 meters and are perforated at each layer. Constant time steps of 10 days up to 10 years are used and the size of the sub-time steps for saturations are based on an estimate of the Courant-Friedrichs-Lewy (CFL) condition[33]. The coarse spaces used for upscaling in AMGe are computed only once at the beginning of the simulation and reused at each time iteration. In addition, since the upscaled saturations are piecewise constant on each agglomerated element, the upscaled problem can be constructed using the local coarse element matrices without visiting the fine grid.

Fig. 7.2 displays the accumulated production of oil and water for the original fine grid reference and the upscaled solutions. From the figure we can see that the upscaling carried out using AMGe is consistently more accurate than the one obtained by traditional flow-based upscaling techniques. Even the coarsest AMGe upscaled solution ($6 \times 22 \times 5$ cells) is more accurate compared to the finer grid upscaled solution ($30 \times 110 \times 5$ cells) for the flow-based method. For many applications, the greater accuracy achieved by the AMGe upscaling technique may justify the higher computational cost of AMGe compared to traditional upscaling techniques. Fig. 7.3 shows the relative upscaling error in the daily production of oil and water. The largest error committed is around 14% for the traditional upscaling, whereas for AMGe the largest error is less than 3%. Table 7.1 displays the number of elements, faces, degrees of freedom, number of non-zeros and complexities for the AMGe upscaling. The arithmetic complexity is defined as the ratio between the total number of degrees of freedom on all levels (fine grid and upscaled) and the fine grid number of degrees of freedom. In a similar way, the operator complexity is the ratio between the total number of non-zeros entries in the mixed system on all levels and the number of non-zeros entries in the fine grid system. Small arithmetic and operator complexities are both highly desirable properties. An arithmetic complexity close to one means a drastic reduction in the size of the problem, while an operator complexity strictly smaller than 2 represents a significant reduction in computational cost.

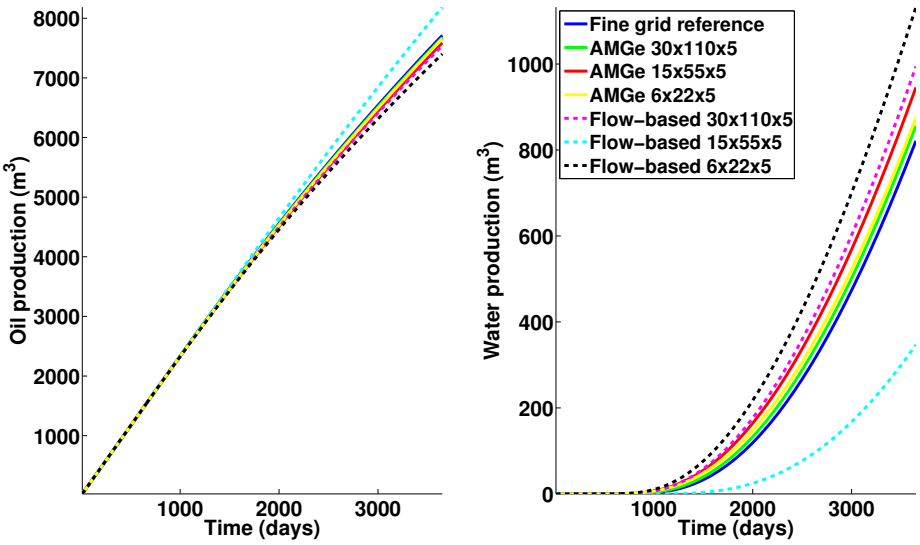


Figure 7.2: Accumulated production.

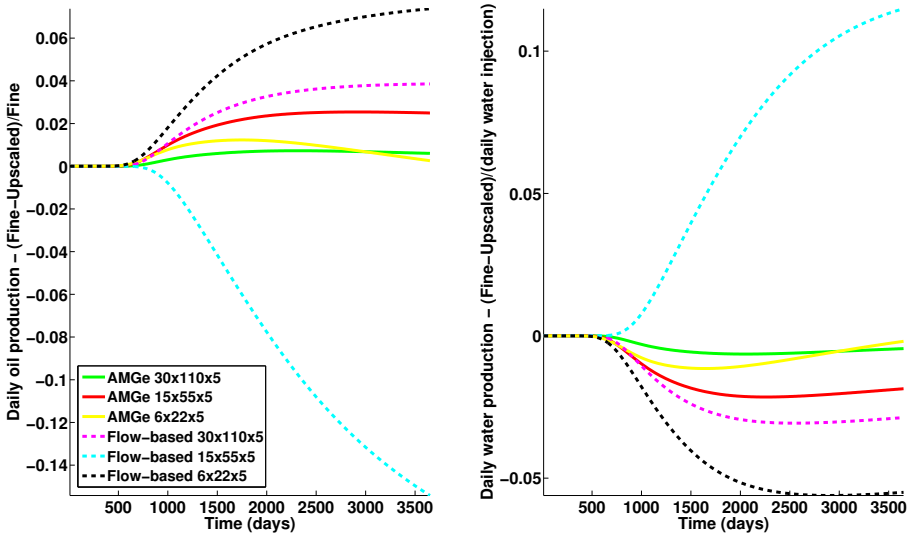


Figure 7.3: Upscaling error for daily production.

Coarse problem size	#elements	#faces	#DoFs	nnz	arithmetic complexity	operator complexity
60 × 220 × 5 (fine)	66000	212600	410600	2984625	-	-
30 × 110 × 5	16500	53500	103100	747325	1.25144	1.25039
15 × 55 × 5	4125	13550	26025	187625	1.06362	1.06286
6 × 22 × 5	660	2252	2252	30797	1.01063	1.01032

Table 7.1: Degrees of freedom, number of non-zeros and complexities for the fine grid and AMGe-upscaled problems.

7.7 Efficient solution of the upscaled problems

In this section we discuss iterative methods for the efficient solution of the upscaled problems. While sparse direct solvers may be a viable solution for small problems, iterative methods are required for large-scale simulation due to the still large size of the upscaled problem. However, traditional state-of-the-art solver which are extremely effective for linear systems arising from finite element discretizations may not perform well for the solution of the upscaled problem. On the contrary, as we demonstrate below, AMGe allows for robust and scalable preconditioning of both the finite element and upscaled linear systems.

In this test, we consider the top 35 layers of the second dataset of the Tenth SPE Benchmark and we use the x-permeability as an isotropic permeability field. The rest of the model parameters are set as in the previous section. Computations are carried out in serial on a cluster with Intel Xeon EP X5660.

The upscaled linear systems are constructed as described in the AMGe section above. First we construct an unstructured agglomerated mesh by using the graph partitioner METIS[66], then we compute the coarse pressure and velocity spaces, and finally we assemble the upscaled linear systems by Galerkin projection of the finite element problem on the coarse spaces. An agglomeration factor of 4, 8, 16, 32, 64 and 128 fine elements per agglomerate is used in the computations to generate upscaled models at different levels of resolution.

To build the AMGe preconditioner, we then recursively apply the upscaling procedure so that a nested hierarchy of agglomerated meshes and coarse spaces is constructed. In this experiment, we recursively apply the agglomeration algorithm with a coarsening factor of 16 “elements” (these are actually agglomerates themselves) per agglomerate until a reasonable small coarsest problem size is achieved (see Fig. 7.4). In addition, to reduce the operator complexity of the AMGe preconditioner and obtain better numerical efficiency (at the cost of a slight increase in the number of iterations), we do not enforce approximation properties for the velocity space at the coarser levels of the AMGe hierarchy.

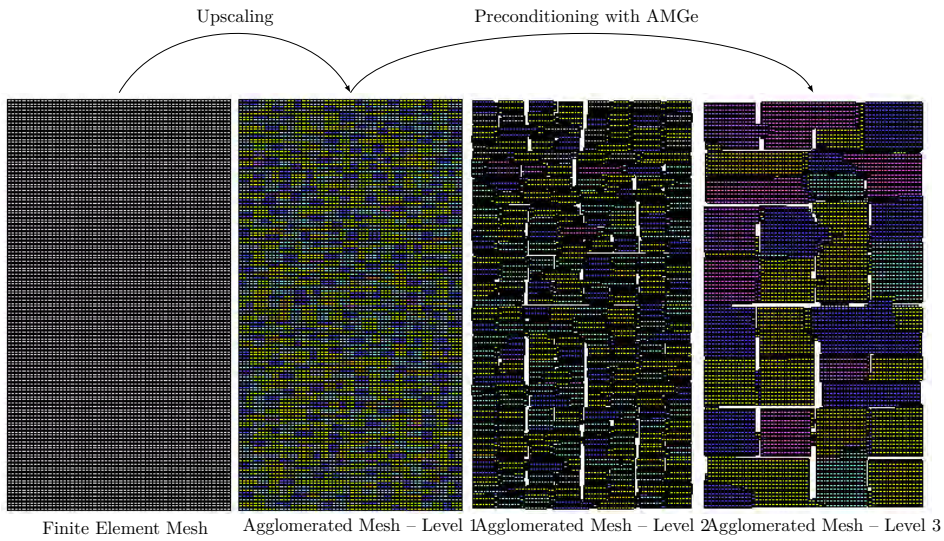


Figure 7.4: Full coarsening of the top 35 layers of the SPE10 benchmark case. The graph partitioner METIS is used. The agglomerated mesh at level 1 consists of roughly 32 fine grid elements per agglomerate and the agglomerated meshes at level 2 and 3 consist of roughly 16 finer grid agglomerates per agglomerate. A coloring algorithm is used to distinguish between agglomerates. Furthermore, agglomerates are slightly shrunk for visualization purposes.

Fig. 7.5 shows the computational time as a function of the number of degrees of freedom. Furthermore, Table 7.2 shows the information about the upscaled systems and the number of iterations to solve the problem with MINRES-$L_2 - H^1$ and GMRES-AMGe. For AMGe, the generation of coarse spaces, the solve itself and the setup cost are reported separately. Since, the coarse spaces depend on the total mobility, which does not necessarily change from time to time in large parts of the domain, we expect a significant reuse of the coarse spaces to be possible. The setup cost entails the factorization of small local matrices so that they can be reused at every GMRES iteration.

We observe that the performance of the $L_2 - H^1$ preconditioner significantly deteriorates for the upscaled problems. The reason of the drastic increase in the number of iterations (for the $L_2 - H^1$ preconditioner) and computational cost (the time to solution for the fine grid problem is almost the same as the one for the coarsest upscaled problem) is that the coarse velocity mass matrix in the upscaled problems is no longer uniformly spectrally equivalent to its diagonal. On the contrary, the AMGe preconditioner shows an optimal convergence behavior

	#elements	#faces	#DoFs	nnz	$L_2 - H^1$ iters.	AMGe iters.
Fine grid problem	462000	1409000	1871000	20813175	64	83
Fine elements per agglomerate						
4	104025	491319	602983	11150935	2593	53
8	57800	287056	360684	7297706	2999	44
16	28905	154599	196481	4370973	4167	66
32	14445	81542	108867	2719543	2039	44
64	7228	45008	70607	2403027	2294	50
128	3613	23626	43573	1885228	1447	31

Table 7.2: Information about the upscaled problem and the number of iterations needed to solve it.

with respect to the number of unknowns, resulting in a significant reduction of the computational cost when solving the upscaled problem.

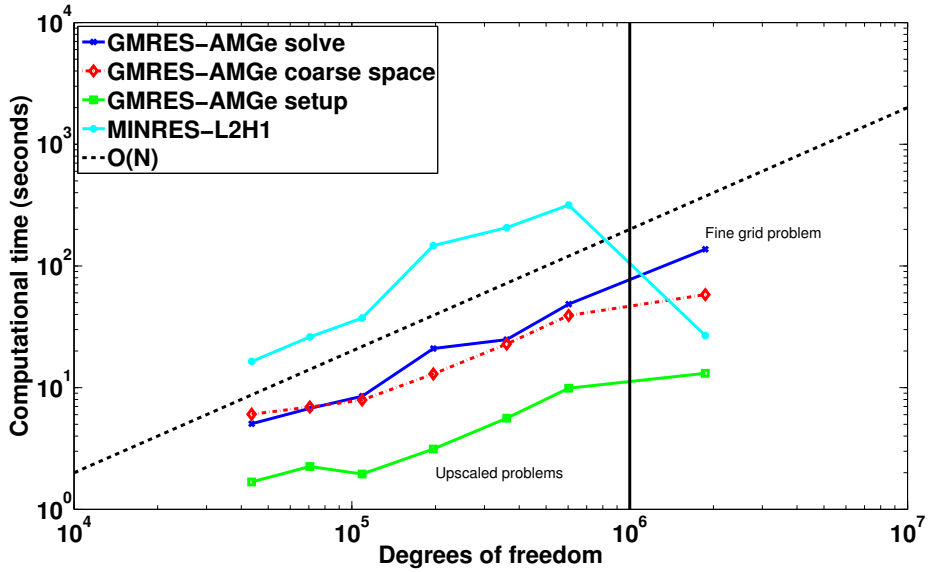


Figure 7.5: Computational time vs degrees of freedom for AMGe and $L_2 - H^1$ preconditioning of the upscaled systems.

7.8 Parallel strong scaling of simulator using the $L_2 - H^1$ preconditioner

In this section, we present preliminary strong scaling results for a distributed memory parallel implementation (in progress) of the reservoir simulator using MPI. For this test, we consider the full SPE10 benchmark using the x-permeability as an isotropic permeability field. Four production wells are placed in each corner and one water injector is placed in the middle of the domain. Both production and injection wells are perforated in all layers. We perform 20 Improved-IMPES steps of length 10 days each. On average, 14 saturation sub-time steps are needed for each Improved-IMPES step. MINRES with $L_2 - H^1$ preconditioner is used to compute the total velocity and pressure at each Improved-IMPES step. The code was executed on *Sierra*, a high performance cluster at Lawrence Livermore National Laboratory consisting of a total of 1,944 nodes connected by Infiniband QDR. Each node of *Sierra* has two 6-cores Xeon EP X5660 Intel CPUs (2.8 Ghz), and 24GB of memory. We use the full capacity of the nodes, i.e. 12 MPI processes per node.

Fig. 10.15 shows the number of MPI processes vs. run time (on the left) and the number of MPI processes vs. speedup (on the right). The measured run-time reported refers to the whole time loop of the simulator, and it includes the computation of rock and fluid properties, the finite element assembly, the set-up of the $L_2 - H^1$ preconditioner, the iterative solution of the mixed systems using MINRES, and the explicit updates of the oil and water saturations. We ran the experiment with a number of MPI processes ranging from 12 to 960, and we compute speedups relative to the measured run-time using 12 processes. We observe a linear speed-up up to 480 processes, which is a highly satisfactory result considering the relatively small size of the problem (approximately 1.1 million elements).

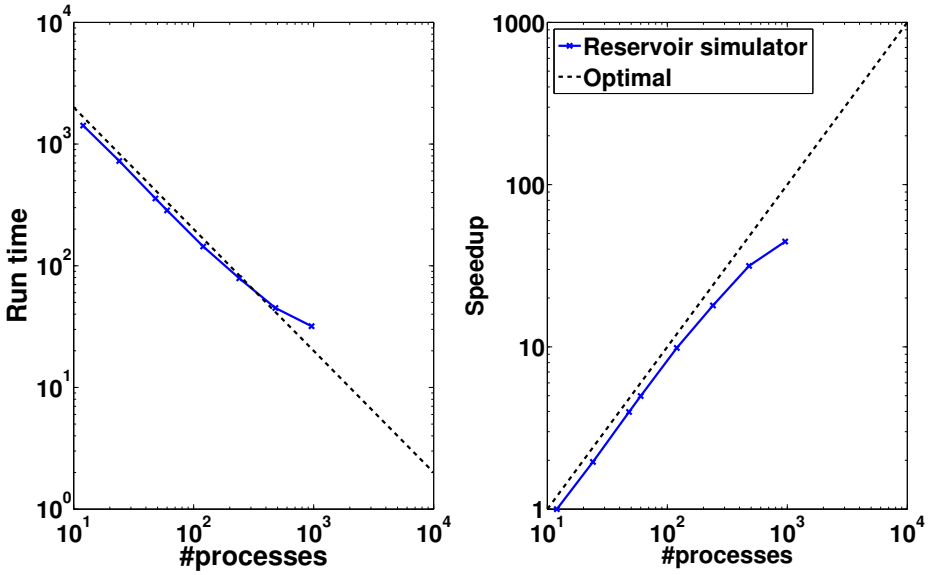


Figure 7.6: Strong scaling of the reservoir simulator for the SPE10 case.

7.9 Conclusions

The effectiveness of the AMGe coarsening techniques both as an upscaling tool and as a preconditioner for GMRES applied to reservoir simulation models was demonstrated. The AMGe upscaling proved to be able to approximate a fine grid reference solution with better accuracy than traditional flow-based upscaling methods. It was also shown that AMGe allows for robust preconditioning of the mixed systems arising from finite element discretization of the flow equations and that it outperforms state-of-the-art solvers for the solution of the algebraically upscaled problems. Finally, the parallel implementation of the reservoir simulator using state-of-the-art algebraic multigrid solvers shows good strong scaling behavior up to hundreds of MPI processes.

Paper IV - Nonlinear Multigrid Solver Exploiting AMGe Coarse Spaces with Approximation Properties

---

Authors: Max la Cour Christensen, Umberto Villa, Allan Peter Engsig-Karup and Panayot Vassilevski.

Presented at 14th Copper Mountain Conference on Iterative Methods, March 20 - March 25, 2016.

Chapter 8

Paper IV - Nonlinear Multigrid Solver Exploiting AMGe Coarse Spaces with Approximation Properties

Abstract: The paper introduces a nonlinear multigrid solver for mixed finite element discretizations based on the Full Approximation Scheme (FAS) and element-based Algebraic Multigrid (AMGe). The main motivation to use FAS for unstructured problems is the guaranteed approximation property of the AMGe coarse spaces that were developed recently at Lawrence Livermore National Laboratory. These give the ability to derive stable and accurate coarse nonlinear discretization problems. The previous attempts (including ones with the original AMGe method, [38, 62]), were less successful due to lack of such good approximation properties of the coarse spaces. With coarse spaces with approximation properties, our FAS approach on unstructured meshes should be as powerful/successful as FAS on geometrically refined meshes. For comparison, Newton's method and Picard iterations with an inner state-of-the-art linear solver is compared to FAS on a nonlinear saddle point problem with applications to porous media flow. It is demonstrated that FAS is faster than Newton's method and Picard iterations for the experiments considered here. Due to the

guaranteed approximation properties of our AMGe, the coarse spaces are very accurate, providing a solver with the potential for mesh-independent convergence on general unstructured meshes.

8.1 Introduction

The Full Approximation Scheme (FAS) is a multigrid method for nonlinear problems, [24, 50, 109, 51]. Its most widespread use is in geometric multigrid on structured grids due to difficulties associated with defining a coarse nonlinear operator on unstructured meshes. On unstructured grids, the most popular choice of nonlinear solver schemes is typically Newton-Krylov methods preconditioned by e.g. a black box method such as Algebraic Multigrid (AMG), [23, 102]. However, FAS offers potential benefits with respect to traditional methods, such as a larger basin of attraction, faster initial convergence, data locality and lower memory footprint. Several papers have addressed the application of FAS to unstructured grids. In [83, 84], FAS based on agglomeration multigrid is compared to Newton-Multigrid. In these papers, coarse grid control-volumes are formed by merging together finer grid control-volumes. Based on this agglomeration of control-volumes, the associated interpolators between grids are defined as simple injection/piecewise constants. In a multilevel context, piecewise constant interpolation between grids is insufficient and will result in loss of accuracy and therefore loss of performance in the overall multigrid scheme, [82]. An improvement was suggested in [82] to use an implicit prolongation operator, however, it may be too expensive to be worth the gain in convergence rate.

This paper can be seen as an extension of the work in [38, 62]. In these papers, FAS is combined with AMGe to obtain a nonlinear solver for lowest order nodal finite elements. Mesh-independent convergence is demonstrated (only) for an elliptic 2D model problem, [38]. The main difference between the work in our paper and [38, 62] is the underlying AMGe method providing the multigrid components, namely the restriction, prolongation and nonlinear coarse operators. In [38], the method is based on the AMGe introduced in [63, 113]. This results in coarse spaces, where only one degree of freedom can be used for each agglomerate. Consequently, it is difficult to maintain accuracy on very coarse agglomerate meshes, resulting in a degradation of the FAS solver performance. The version of AMGe used in this paper, [73, 72], allows the construction of operator-dependent coarse spaces for the whole de Rham complex (i.e. the sequence of H^1-conforming, $H(curl)$-conforming, $H(div)$-conforming and L^2-conforming spaces). This gives the foundation to cover a broad range of applications such as elliptic PDEs, Maxwell equations, Darcy flow equations, etc. In this paper, we restrict ourselves to the use the $H(curl)$, $H(div)$ and L^2 spaces.

The recently developed AMGe technique with guaranteed approximation properties on coarse agglomerated meshes provides the coarse spaces used for the restriction, prolongation and nonlinear operators. These coarse spaces have the properties necessary (and are intended) to be used as an upscaling tool, but in this paper we demonstrate that the same coarse spaces can be reused for solvers. The coarse spaces have desirable properties analogous to the original finite element spaces: Nédélec, Raviart-Thomas and discontinuous piecewise. This is ensured by introducing additional degrees of freedom associated with non-planar interfaces/edges between coarse elements/faces (agglomerates of finer level elements/faces). In this way, the necessary number of degrees of freedom on coarse faces or coarse edges are automatically found via singular value decomposition.

The FAS-AMGe method implemented in this paper is tested on a nonlinear saddle point problem with applications in porous media flow. It is compared to exact and inexact Newton's method and Picard iterations. The comparison is done in a fair way by letting the FAS, Newton's method and Picard iterations utilize the same underlying components, namely the multilevel divergence free solver, recently developed at LLNL, for the solution of the mixed discretization of the Darcy problem.

The rest of the paper is structured as follows. In Section 8.2 we give a brief outline of the AMGe method we use. Section 8.3 summarizes the FAS we use in general terms. The model problem of our main interest is introduced in Section 8.4. A key ingredient of our solver, namely, a divergence-free preconditioner is briefly summarized in Section 8.5. The main part of this paper consisting of a large set of numerical tests, is given in Section 8.6. At the end, in Section 8.7 we provide some conclusions and perspectives.

8.2 Element-based Algebraic Multigrid (AMGe)

AMGe is a framework of multilevel methods for the solution of systems stemming from finite element discretizations. In contrast to AMG, where only system coefficients are used, AMGe also employs grid topology and finite element matrices. The specific version of AMGe used in this paper was introduced in [73, 72, 99]. The method facilitates the construction of operator-dependent coarse spaces which can be shown to guarantee approximation on coarse levels for general unstructured meshes. Thanks to the guaranteed approximation properties, this AMGe technique can be used as a discretization tool (upscaling) on coarse (agglomerated) meshes and allows for the generation of accurate coarse spaces for the FAS hierarchy.

In a setup phase, a hierarchy of agglomerated meshes is formed. Each agglomerate is formed by grouping together finer-grid elements (or agglomerates if already on a coarse level). For unstructured meshes, the agglomeration can be accomplished by the use of graph partitioners. In particular, this work uses METIS, [66], to form agglomerates. Once the hierarchy of agglomerated meshes is generated, coarse spaces are computed by restricting certain basis functions and by solving local saddle point problems for each agglomerate entity. A thorough description of the methods involved is out of scope for this paper. In addition, this version of AMGe allows to assemble the coarse grid residuals and Jacobians directly on coarse agglomerated meshes without visiting the fine grid. For details on the assembly procedure see [33], where the time-dependent two-phase porous media flow (reservoir simulation) is solved with optimal complexity on coarse (upscaled) levels. The software developed for this paper uses the Element-Agglomeration Algebraic Multigrid and Upscaling Library: ParElag developed at Lawrence Livermore National Laboratory. ParElag is based on the MFEM library, [2], for the finite element discretization and supports several solvers from the HYPRE library, [53].

8.3 Full Approximation Scheme (FAS)

The FAS, [24, 50, 109], can be considered as a generalization of multigrid methods to nonlinear problems. For a two-grid method, consider the nonlinear dicrete problem:

$$\mathcal{A}_h(u_h) = f_h, \tag{8.1}$$

where $\mathcal{A}_h$ is a nonlinear operator, and the subscript h indicates that all quantities are discretized on the fine grid. Introducing the approximate solution v_h, the residual equation is given by

$$\mathcal{A}_h(u_h) - \mathcal{A}_h(v_h) = r_h, \tag{8.2}$$

where $r_h = f_h - \mathcal{A}_h(v_h)$. Introducing the subscript H to refer to quantities defined on the coarse mesh, the coarse residual equation can be written as

$$\mathcal{A}_H(u_H) - \mathcal{A}_H(v_H) = r_H \;\Leftrightarrow\; \mathcal{A}_H(v_H + e_H) - \mathcal{A}_H(v_H) = r_H, \tag{8.3}$$

where e_H is the error $u_H - v_H$.

To restrict the fine quantities v_h and e_h, we use the projection operator $\Pi : h \to H$, while to restrict the residual r_h to the coarse grid we use the transpose of the prolongation operator $P : H \to h$. The operators P and Π are constructed

by our AMGe algorithm such that $\Pi P = I_H$. More specifically, the coarse grid problem reads

$$\mathcal{A}_H \underbrace{(\Pi v_h + \Pi e_h)}_{u_H} = \underbrace{\mathcal{A}_H(\Pi v_h) + P^T r_h}_{f_H}, \tag{8.4}$$

The coarse grid correction is then given by $e_H = u_H - \Pi v_h$. This correction term is prolongated to the fine grid level by using the prologation operator P and the solution v_h is updated accordingly. Algorithm 5 contains a pseudo code for the multilevel implementation of a FAS V-cycle.

Algorithm 4 Pseudo code for FAS V-cycle implementation.
Inputs:
Approximate solution: $u_{l=0}$
Nonlinear operator: $\mathcal{A}_l$, $l = 0, \ldots,$ nLevels
Right hand side: $f_{l=0}$
Output:
Approximate solution $u_{l=0}$

```
1: function FAS_Vcycle(l)
2:     if l == nLevels-1 (coarsest grid) then
3:         Approximately solve A_l(u_l) = f_l
4:     else
5:         Nonlinear smoothing of A_l(u_l) = f_l
6:         Compute defect: d_l = f_l - A_l(u_l)
7:         Restrict defect: d_{l+1} = P^T d_l
8:         Restrict solution: u_{l+1} = Π u_l
9:         Store approximate solution: u_old = u_{l+1}
10:        Compute right hand side for residual equation: f_{l+1} = d_{l+1} +
   A_{l+1}(u_{l+1})
11:        Apply FAS_Vcycle(l+1) to compute updated u_{l+1}
12:        Compute correction: v_{l+1} = u_{l+1} - u_old
13:        Prolongate correction: v_l = P v_{l+1}
14:        Correct approximation: u_l = u_l + v_l
15:        Nonlinear smoothing of A_l(u_l) = f_l
16:    end if
17: end function
```

8.4 Model problem

The saddle point problem of focus in this paper is chosen for the simplicity of the formulation, while still presenting itself as a numerically challenging nonlinear problem with applications in porous media flow. The problem stems from Darcy's law and reads

$$\begin{cases} k^{-1}(p)\mathbf{u} + \nabla p = f_u \\ \nabla \cdot \mathbf{u} = f_p, \end{cases} \tag{8.5}$$

where k is the conductivity field or permeability field as it is commonly called in petroleum engineering, p is the pressure and $\mathbf{u}$ is the velocity. The pressure dependency of the permeability is modeled as

$$k(p) = k_0 e^{-\alpha p}, \tag{8.6}$$

where k_0 is a user-given permeability at reference pressure 0. The problem can be used to model a steady-state single phase primary depletion of an oil reservoir, where the permeability decreases exponentially with the pressure.

For simplicity, we assume essential boundary conditions $\mathbf{u} \cdot \mathbf{n} = 0$, but non-homogeneous and natural boundary conditions can be handled in a similar way.

8.4.1 Notation

We now introduce some notation used throughout the paper. Let Ω be a bounded connected domain in $\mathbb{R}^d$ with a regular (Lipschitz continuous) boundary $\partial\Omega$, which has a well-defined unit outward normal vector $\mathbf{n} \in \mathbb{R}^d$. For the cases considered in this paper, $d = 3$. For the vectorial functions $\mathbf{u}, \mathbf{v} \in \mathbf{L}^2(\Omega) = [L^2(\Omega)]^d$ and scalar functions $p, w \in L^2(\Omega)$, we define the inner products $(\mathbf{u}, \mathbf{v}) = \int_\Omega \mathbf{u} \cdot \mathbf{v}\, d\Omega$ and $(p, w) = \int_\Omega p\, w\, d\Omega$. Finally, we introduce the functional space $H(\mathrm{div}; \Omega)$ defined as

$$H(\mathrm{div}; \Omega) := \{\mathbf{u} \in \mathbf{L}^2(\Omega) \mid \mathrm{div}\, \mathbf{u} \in L^2(\Omega)\}.$$

Using the above notation, we define the functional spaces $\mathcal{R}$ and $\mathcal{W}$ as

$$\mathcal{R} \equiv \{\mathbf{u} \in H(\mathrm{div}; \Omega) \mid \mathbf{u} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega\};$$
$$\mathcal{W} \equiv L^2(\Omega).$$

8.4.2 Weak formulation

To derive the weak formulation for the mixed system in equations (8.5), we multiply equations (8.5) with the test functions $\mathbf{v} \in \mathcal{R}$ and $w \in \mathcal{W}$ and integrate over the domain Ω. After integration-by-parts of the non-conforming terms and applying the no-flux boundary condition $\mathbf{u}\cdot\mathbf{n} = \mathbf{v}\cdot\mathbf{n} = 0$, we obtain the following variational problem

PROBLEM 11 *Find* $(\mathbf{u}, p) \in \mathcal{R} \times \mathcal{W}$ *such that*

$$\begin{cases} \Big(k(p)^{-1}\mathbf{u}, \mathbf{v}\Big) - \Big(p, \nabla\cdot\mathbf{v}\Big) = (f, \mathbf{v})\,, & \forall\ \mathbf{v} \in \mathcal{R} \\ \Big(\nabla\cdot\mathbf{u}, w\Big) = (q, w)\,, & \forall\ w \in \mathcal{W} \end{cases}$$

To solve the non-linear Problem 11 we consider both the Newton's and Quasi-Newton's (Picard) methods. In a compact notation, the Newton's/Picard's step reads

$$\begin{aligned} \text{Solve:}\quad & a(\delta\mathbf{u}, \delta p; \mathbf{v}, w) = -r(\mathbf{u}_{\text{old}}, p_{\text{old}}; \mathbf{v}, w), \quad \forall (\mathbf{v}, \delta w) \in \mathcal{R}\times\mathcal{W}; \\ \text{Update:}\quad & \mathbf{u}_{\text{new}} = \mathbf{u}_{\text{old}} + \delta\mathbf{u}, \quad p_{\text{new}} = p_{\text{old}} + \delta p, \end{aligned} \tag{8.7}$$

where the residual variational form is

$$\begin{aligned} r(\mathbf{u}_{\text{old}}, p_{\text{old}}; \mathbf{v}, w) = \Big(k(p_{\text{old}})^{-1}\mathbf{u}_{\text{old}}, \mathbf{v}\Big) - \Big(p_{\text{old}}, \nabla\cdot\mathbf{v}\Big) - (f, \mathbf{v}) \\ + \Big(\nabla\cdot\mathbf{u}_{\text{old}}, w\Big) - (q, w), \quad \forall (\mathbf{v}, w) \in (\mathcal{R}, \mathcal{W}), \end{aligned} \tag{8.8}$$

and the bilinear form for the Jacobian (Approximate Jacobian) evaluated at $(\mathbf{u}_{\text{old}}, p_{\text{old}})$ is

$$\begin{aligned} a(\delta\mathbf{u}, \delta p; \mathbf{v}, w) = \Big(k(p_{\text{old}})^{-1}\delta\mathbf{u}, \mathbf{v}\Big) + \beta\cdot\Big(\frac{\partial k^{-1}}{\partial p}\Big|_{p=p_{\text{old}}}\mathbf{u}_{\text{old}}\delta p, \mathbf{v}\Big) \\ - \Big(\delta p, \nabla\cdot\mathbf{v}\Big) - (f, \mathbf{v}) + \Big(\nabla\cdot\delta\mathbf{u}, w\Big) - (q, w), \quad \forall (\mathbf{v}, w) \in (\mathcal{R}, \mathcal{W}). \end{aligned} \tag{8.9}$$

Here $\beta = 0$ leads to Picard iterations and $\beta = 1$ leads to Newton's method.

8.4.3 Mixed Finite Element Discretization

The variational non-linear problem 11 and its linearization in (8.7) is discretized with the Mixed Finite Element method. In particular, we let $\mathcal{R}_h \subset \mathcal{R}$ be the

(lowest order) Raviart–Thomas finite element space consisting of vector functions with a continuous normal component across the interfaces between the elements and $\mathcal{W}_h \subset \mathcal{W}$ be the space of piecewise discontinuous polynomials (constant) scalar functions. It is well-known that this choice of finite element spaces satisfies the Ladyzhenskaya-Babuška-Brezzi conditions, and therefore allows for a stable discretization.

To obtain the discrete version of (8.7), let us denote with $\left\{\boldsymbol{\phi}^j\right\}_{j=1,\dots,\dim(\mathcal{R}_h)}$ a basis for the space $\mathcal{R}_h$ and $\left\{\psi^j\right\}_{j=1,\dots,\dim(\mathcal{W}_h)}$ a basis for the space $\mathcal{W}_h$. With this notation, the finite element solution $(\mathbf{u}_h, p_h)$ can be written as a linear combination of the basis functions $(\boldsymbol{\phi}^j, \psi^j)$. More specifically, letting $\mathbf{U} \in \mathbb{R}^{\dim(\mathcal{R}_h)}$ and $P \in \mathbb{R}^{\dim(\mathcal{W}_h)}$ denote the vectors collecting the finite element degrees of freedom $\mathbf{u}_h^i$, $i = 1, \dots, \dim(\mathcal{R}_h)$ and p_h^i, $i = 1, \dots, \dim(\mathcal{W}_h)$, we write

$$\mathbf{u}_h = \sum_{j=1}^{\dim(\mathcal{R}_h)} \mathbf{u}_h^j \boldsymbol{\phi}^j, \qquad p_h = \sum_{j=1}^{\dim(\mathcal{W}_h)} p_h^j \psi^j. \tag{8.10}$$

We introduce the finite element matrices M, B and N whose entries are given by

$$\begin{aligned}
M_{ij} &= \left(k^{-1}(p_{h,\mathrm{old}})\boldsymbol{\phi}^j, \boldsymbol{\phi}^i\right), && i,j = 1,\dots,\dim(\mathcal{R}_h),\\
B_{ij} &= \left(\nabla\cdot\boldsymbol{\phi}^j, \psi^i\right), && i = 1,\dots,\dim(\mathcal{W}_h), j = 1,\dots,\dim(\mathcal{R}_h),\\
N_{ij} &= \left(\left.\tfrac{\partial k^{-1}}{\partial p}\right|_{p=p_{h,\mathrm{old}}} \mathbf{u}_{h,\mathrm{old}}\psi^j, \boldsymbol{\phi}^i\right), && i = 1,\dots,\dim(\mathcal{R}_h), j = 1,\dots,\dim(\mathcal{W}_h).
\end{aligned} \tag{8.11}$$

The Galerkin formulation leads to the solution of the sparse linear system

$$\mathbf{A}\mathbf{X} = \mathbf{B}, \tag{8.12}$$

where the block matrix $\mathbf{A}$ and block vectors $\mathbf{X}$ and $\mathbf{B}$ read:

$$\mathbf{A} = \begin{bmatrix} M & B^T + \beta\cdot N \\ B & 0 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} \delta\mathbf{U} \\ \delta P \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{F} \\ Q \end{bmatrix}, \tag{8.13}$$

where $\beta = 0$ will lead to Picard iterations and $\beta = 1$ will lead to Newton's method.

8.4.4 Multilevel formulation

Using the AMGe technique in Section 8.2, we construct coarse spaces $\mathcal{R}_l$ and $\mathcal{W}_l$ for each level of the hierarchy l. The *inf-sup compatibility* of the coarse spaces

is a direct consequence of the compatibility of the fine grid spaces $\mathcal{R}_0$ and $\mathcal{W}_0$ and of the commutativity of the diagram

$$\begin{array}{ccc} \mathcal{R}_l & \xrightarrow{D_l} & \mathcal{W}_l \\ \Pi_l^{\mathcal{R}} \downarrow \uparrow P_l^{\mathcal{R}} & & \Pi_l^{\mathcal{W}} \downarrow \uparrow P_l^{\mathcal{W}} \\ \mathcal{R}_{l+1} & \xrightarrow{D_{l+1}} & \mathcal{W}_{l+1} \end{array} ,$$

where, by commutativity, we mean that the identities $\Pi_l^{\mathcal{W}} D_l = D_{l+1}\Pi_l^{\mathcal{R}}$ and $D_l P_l^{\mathcal{R}} = P_l^{\mathcal{W}} D_{l+1}$ hold. Here the $D_i : \mathcal{R}_i \to \mathcal{W}_i$ $(i = l, l+1)$ is the discrete operator representing the mapping $D_i \mathbf{u}_i = \operatorname{div} \mathbf{u}_i \in \mathcal{W}_i$ for all $\mathbf{u}_i \in \mathcal{R}_i$; $P_l = [P_l^{\mathcal{R}}; P_l^{\mathcal{W}}] : (\mathcal{R}_{l+1}, \mathcal{W}_{l+1}) \to (\mathcal{R}_l, \mathcal{W}_l)$ is the prolongation operator from coarse to fine, and $\Pi_l = [\Pi_l^{\mathcal{R}}; \Pi_l^{\mathcal{W}}] : (\mathcal{R}_l, \mathcal{W}_l) \to (\mathcal{R}_{l+1}, \mathcal{W}_{l+1})$ is the projection operator.

Finally, to apply the FAS V-cycle, we let x_l be the unknowns $(\mathbf{u}_l, p_l)$, we define the nonlinear differential operator $\mathcal{A}_l : (\mathcal{R}_l, \mathcal{W}_l) \to (\mathcal{R}_l^*, \mathcal{W}_l^*)$ as

$$\langle A_l(x_l), y_l \rangle = r_l(\mathbf{u_l}, p_l; \mathbf{v}_l, w_l), \quad \forall y_l = (\mathbf{v}_l, w_l) \in (\mathcal{R}_l, \mathcal{W}_l),$$

and, on the fine grid, we set $f_0 = 0$ to match (8.7).

8.5 Multilevel Divergence Free preconditioner

Each Newton/Picard step require the solution of a linear system of the form (8.12), where the matrix $\mathbf{A}$ is indefinite (saddle point problem). We use the GMRES method preconditioned by a specialized indefinite AMGe preconditioner (the Multilevel Divergence Free preconditioner - MLDivFree) developed at LLNL for the solution of the mixed formulation of the Darcy equations. More specifically, MLDivFree uses a hierarchy of AMGe coarse spaces to form a preconditioner for symmetric indefinite saddle point problems of the form in (8.12) (when $\beta = 0$). MLDivFree can be summarized in the following three actions:

1. Find $\hat{u}$ such that the divergence constraint $B\hat{u} = q$ is satisfied.
2. Find $u = \hat{u} + C\sigma$ such that $||M(\hat{u} + C\sigma) - f||^2_{M^{-1}} \to \min$, where C is the discretization of the curl operator (also obtained by AMGe).
3. Find p such that $||B^T p - Mu - f||^2_{M^{-1}} \to \min$. This is the dual operation of step 1.

In practice, this is implemented by a symmetric V-cycle with a sophisticated multiplicative smoother. The pre-smoothing involves first solving for each agglomerate a local saddle point problem. Next a divergence free correction is obtained by solving for $\delta u = C(\delta\sigma)$, where $\sigma \in H(curl)$ is computed by applying some smoothing iteration to the linear system $C^T M C\sigma = C^T f$. The post-smoother consists of the same two components but in the reverse order.

In the numerical results section, to allow for a fair comparison among all the nonlinear solvers, we will apply MLDivFree both as a preconditioner of the linear systems in Newton's and Picard's method and in the smoothing phase of FAS.

8.6 Numerical results

In this section, scaling experiments are carried out for a structured hexahedral mesh and for an unstructured tetrahedral mesh. For the exact Newton's method and Picard iterations we solve the linear system using preconditioned GMRES up to a relative tolerance of 10^{-8} and absolute tolerance of 10^{-10}. The inexact Newton's method and Picard iterations are based on a Eisenstat-Walker type condition, [42], to determine the relative tolerance for the linear solver GMRES in each nonlinear iteration and prevent *oversolving*. We use the following expression: $\min(0.5, \sqrt{||r^k||_2/||r^0||_2})$. Globalization of the Newton/Picard method is achieved by backtracking. The stopping criterion of the nonlinear solvers used for all experiments is $||r^k||_2 \leq \max(\text{rtol}||r^0||_2, \text{atol})$, where $\text{rtol} = 10^{-6}$ and $\text{atol} = 10^{-8}$ are the relative and absolute tolerances. Furthermore, in this section, we use two measures of multigrid performance, namely the arithmetic complexity and the operator complexity. The arithmetic complexity C_a is defined as the ratio of the total number of degrees of freedom on all levels (fine grid and coarse) to the fine grid number of degrees of freedom. In a similar way, the operator complexity C_o is the ratio of the total number of non-zeros (in the mixed system) on all levels to the number of non-zeros on the fine grid. More specifically, we have

$$C_a = \frac{\sum_{l=0}^{\text{levels-1}} \dim(\mathcal{R}_l \times \mathcal{W}_l)}{\dim(\mathcal{R}_0 \times \mathcal{W}_0)} \qquad C_o = \frac{\sum_{l=0}^{\text{levels-1}} \text{nnz}(\mathbf{A}_l)}{\text{nnz}(\mathbf{A}_0)}. \tag{8.14}$$

We stress upon the fact that many methods in practice can achieve C_a close to unity and have acceptable approximation properties. However, it is also of vital importance to ensure that C_o is small (at least sufficiently less than two) since then the coarse systems take up much less memory than the fine grid problem.

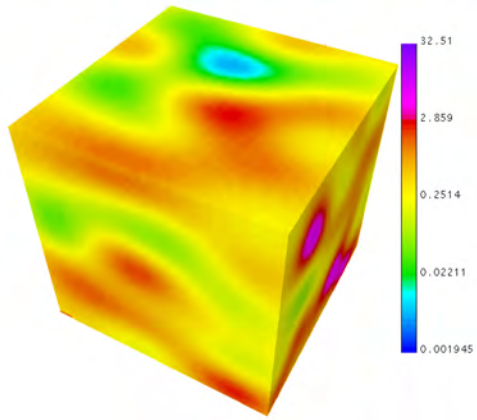


Figure 8.1: Permeability field used for scaling experiments. Generated by a truncated Karhunen-Loève expansion.

8.6.1 Structured grid scaling

The first study is a comparison between FAS, Newton's method and Picard iterations on a structured grid. The computational domain is the unit cube, discretized with a structured cartesian hexahedral mesh. The permeability coefficient k_0 is the realization of a lognormal spatially correlated random field displayed in Figure 8.1. It is generated by means of a truncated Karhunen-Loève (KL) expansion with 6 eigenmodes in each direction, a standard deviation of 3 and a correlation length of 0.1. The KL expansion is chosen for its ability to generate a grid-independent permeability field such that scaling experiments can be easily performed. Furthermore, it does have some similarities with actual permeability fields. The parameter α is set to 10.

Five different solver schemes are compared, namely

- (1) FAS with Picard linearization. Each smoothing step is using one V-cycle of the MLDivFree preconditioner.
- (2) Exact and (3) inexact Picard iterations with GMRES preconditioned by MLDivFree.
- (4) Exact and (5) inexact Newton's method with GMRES preconditioned by MLDivFree. Note that for preconditioning, MLDivFree uses the symmetric matrix stemming from the Picard linearization.

Figure 8.2 shows the computational time as a function of degrees of freedom

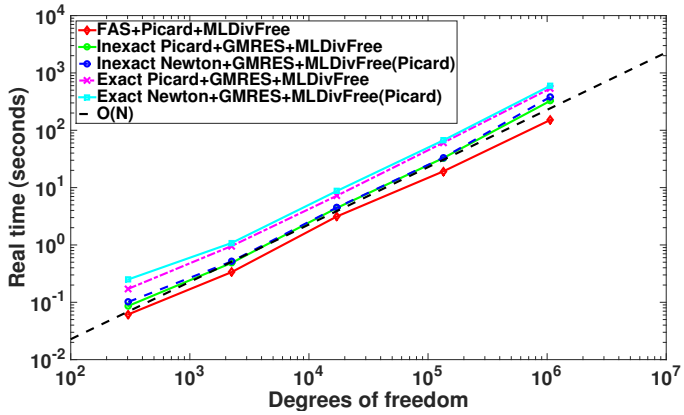


Figure 8.2: Computational time for various solver schemes as a function of problem size (structured grid).

for all solver schemes. The hierarchy of agglomerated meshes is structured (cartesian) and with a coarsening factor of 2 in each direction. The coarsest level is 1 agglomerate and the number of multigrid levels range from 3 (on the coarsest initial mesh) to 7 (on the finest initial mesh) for these experiments. It is evident that all solver schemes have mesh-independent convergence for the given problem. The inexact solvers are faster than the exact solvers and FAS is the fastest overall. Table 8.1 holds more information on the results.

We notice that, for this particular problem, the Picard's method converges in nearly the same number of iterations as Newton's method. We suspect that the suboptimal convergence of Newton is due to the fact that $\alpha = 10$ results in very small basin of attraction for the Newton's method and that backtracking is needed to ensure global converge of the Newton method.

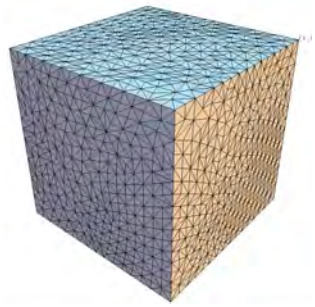
8.6.2 Unstructured grid scaling

In this section, we carry out a scaling experiment for an unstructured tetrahedral mesh. A unit cube is meshed with NETGEN, [94, 106], to produce 8 unstructured meshes with increasing resolution. We use the same permeability coefficient k_0 and parameter α as in the structured test case. For all 8 meshes, the performance of FAS is compared to the performance of inexact Picard. We restrict ourselves to these two nonlinear methods, since they have proven to be the fastest. The agglomeration is carried out using the graph partitioner METIS with a post-processing to ensure certain topological requirements are met. The

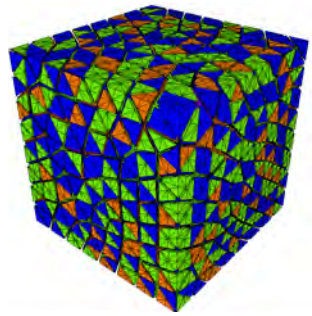
FAS+Picard+MLDivFree	#elements	#linears	#nonlinears	time/time(FAS)
	64	-	5	-
	512	-	4	-
	4096	-	5	-
	32768	-	4	-
	262144	-	4	-
Inexact Picard+GMRES+MLDivFree				
	64	12	9	1.41
	512	11	7	1.45
	4096	13	9	1.41
	32768	12	9	1.71
	262144	12	10	2.16
Inexact Newton+GMRES+MLDivFree(Picard)				
	64	16	9	1.65
	512	12	7	1.54
	4096	13	9	1.43
	32768	12	9	1.73
	262144	12	10	2.52
Exact Picard+GMRES+MLDivFree				
	64	37	9	2.79
	512	33	6	2.85
	4096	34	7	2.30
	32768	37	7	3.17
	262144	35	7	3.55
Exact Newton+GMRES+MLDivFree(Picard)				
	64	59	9	4.07
	512	38	6	3.22
	4096	41	7	2.78
	32768	42	7	3.48
	262144	44	7	3.96

#elements	#DoFs	operator complexity	arithmetic complexity	#levels
64	304	1.32	1.17	3
512	2240	1.32	1.16	4
4096	17152	1.31	1.15	5
32768	134144	1.25	1.15	6
262144	1060864	1.18	1.15	7

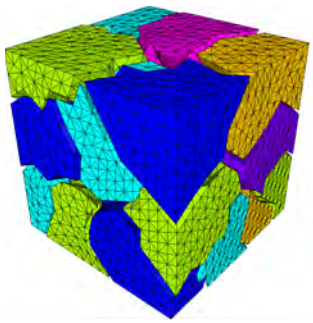
Table 8.1: Information on the structured grid scaling experiments. #linears is the total number of linear iterations for all nonlinear iterations.



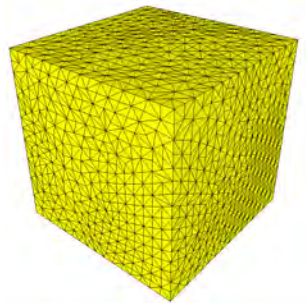
(a) Level 0
Uniformly refined once



(b) Level 1
Agglomerates are based on the uniform refinement



(c) Level 2
Agglomerates are formed by METIS



(d) Level 3
Coarsest level is one agglomerate

first level of the mesh hierarchy is geometric (i.e. mesh derefinement) and the following levels are algebraic (i.e. METIS). The initial geometric level allows for smaller operator complexity. Figures 8.3a - 8.3d give an example of the topology produced by the procedure. The unstructured coarsening factor (METIS) used in these experiments is 100 finer elements per agglomerated element.

From Table 8.2, it can be seen that both algorithms perform optimally in terms of linear and nonlinear iterations. If we look at Figure 8.3, the computational time is good, but slightly suboptimal due to the operator complexity not remaining constant. This can be remedied by increasing the coarsening factor for the larger problem sizes.

FAS+Picard+MLDivFree	Experiment	#linears	#nonlinears	time/time(FAS)
	32280	-	5	-
	53832	-	4	-
	67624	-	4	-
	215512	-	4	-
	405632	-	4	-
	496800	-	4	-
	679808	-	4	-
	827144	-	4	-
Inexact Picard+GMRES+MLDivFree				
	32280	13	9	1.42
	53832	13	9	1.66
	67624	13	9	1.59
	215512	11	9	1.33
	405632	11	9	1.20
	496800	12	9	1.32
	679808	12	10	1.28
	827144	13	9	1.18

#elements	#DoFs	operator complexity	arithmetic complexity	#levels
32280	98644	1.19	1.135	4
53832	164268	1.21	1.135	4
67624	207056	1.21	1.136	4
215512	654332	1.23	1.136	5
405632	1227916	1.29	1.140	5
496800	1507968	1.34	1.144	5
679808	2066320	1.34	1.140	5
827144	2500552	1.34	1.140	5

Table 8.2: Information on the unstructured grid scaling experiments. #linears is the total number of linear iterations for all nonlinear iterations.

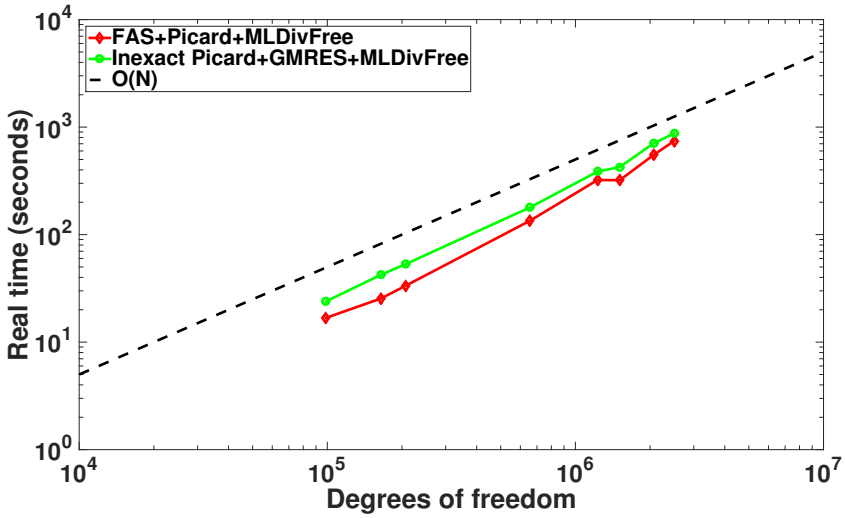


Figure 8.3: Computational time for FAS and inexact Picard as a function of problem size (unstructured grids generated by NETGEN).

8.7 Conclusion & perspectives

AMGe with guaranteed approximation properties has been combined with FAS to demonstrate a scalable nonlinear solver for a challenging saddle point problem. Numerical tests have been performed demostrating the mesh independent convergences of FAS (number of V-cycles). We compared FAS-AMGe to exact and inexact Newton's method and Picard iterations; FAS outperformed the exact methods (4X faster) and also proved slightly faster than the inexact versions. In this paper, only global linearization has been considered. A more thorough study comparing with local linearization techniques would be interesting.

Chapter 9

Multilevel Monte Carlo using AMGe

A great application of AMGe is for the construction of numerically upscaled models to be used for the acceleration of uncertainty quantification. As it is well know, the Monte Carlo method can be used to estimate the mean and variance for some quantity of interest. In particular, the standard Monte Carlo estimator for the expected value is

$$\bar{Q} = \frac{1}{N} \sum_{i=1}^{N} Q^i, \tag{9.1}$$

where Q^i is the i^{th} sample of a random variable and N is the number of independent samples. The variance of the Monte Carlo estimator is given by

$$\frac{1}{N} \sum_{i=1}^{N} (Q^i - \bar{Q}), \tag{9.2}$$

Having the ability to estimate the mean and variance of some important quantity of interest for a particular model is a very useful tool. It can provide engineers and decision makers with a much better understanding and characterization of the uncertainties involved in their problem. These uncertainties are often introduced by poor model data, but many other circumstances may introduce these.

The biggest problem with the Monte Carlo method is the fact that it often entails many realizations of the given model. In many cases and in particular when the underlying model is a 3D discretization of a partial differential equation (or a system of partial differential equations), the model is simply too computationally expensive to be able to afford such a large number of realizations. This makes the Monte Carlo method infeasible.

An excellent way to alleviate this issue was the introduction of the Multilevel Monte Carlo method, [46]. In the Multilevel Monte Carlo method, the Monte Carlo process is accelerated by the realization of coarser models. Typically a hierarchy of coarser models is used. This is where AMGe can play a major role, since it allows for the construction of computationally cheaper models with good accuracy. The Multilevel Monte Carlo method relies on a multilevel decomposition. For example, the expected value of some quantity of interest can be computed as

$$E[Q_0] = E[Q_L] + \sum_{l=1}^{L} E[Q_{l-1} - Q_l], \tag{9.3}$$

where 0 indicates the finest grid. From the formula, it is evident that instead of carrying out realizations of the model solely on the finest grid, now the model is evaluated on L levels. Borrowing from the basic multigrid idea, the difference between the evaluated quantities of interest on two levels is used to compute the expected value. For Multilevel Monte Carlo, the mean square error is given by

$$\text{MSE} = \frac{1}{N_l}\text{Var}[Q_L] + \sum_{l=1}^{L} \text{Var}[Q_{l-1} - Q_l] + (E[Q - Q_0])^2, \tag{9.4}$$

The first term of (9.4) is on the coarsest level L, hence fixed. The intermediate terms have the property that $\text{Var}[Ql - 1 - Q_l] \ll \text{Var}[Q_l]$, hence require much less samples. The last term is the fine grid discretization error.

In practice, the Multilevel Monte Carlo method works by first carrying out a number of realizations of the model on the coarsest level. When a satisfactory amount of realizations have been made, the method moves to the second finest level and so forth.

9.1 Numerical results

In the following, the Multilevel Monte Carlo method is applied as in [36]. To test the efficiency of the method, a numerical study is carried out. The study

uses the reservoir simulation code developed for Paper II. The model is the two-phase incompressible fluids and rock (mixed) formulation as given by

$$\mathbf{K}^{-1}\lambda^{-1}(S)\mathbf{u} + \nabla p = \left(\sum_{\alpha} \rho_\alpha f_\alpha(S)\right) g\nabla z \tag{9.5}$$

$$\nabla \cdot \mathbf{u} = q(S,p) \tag{9.6}$$

$$\phi\frac{\partial S_\alpha}{\partial t} + \nabla \cdot \mathbf{u}_\alpha(S,\mathbf{u}) = \frac{q_\alpha(S,p)}{\rho_\alpha}, \tag{9.7}$$

For a complete description of this model, we refer to Paper II. The specific case investigated here is a segment of the SPE10 model, [95]. Specifically, the top layer of the permeability field is used and the location of the injection and production wells. The injection well is located in the middle of the domain (see Figure 9.1) and the four production wells are located in each corner.

We consider the permeability field as a stochastic variable and apply a Truncated Karhunen-Loève expansion to generate stochastic samples. See Figure 9.1 for an illustration of the stochastic sample.

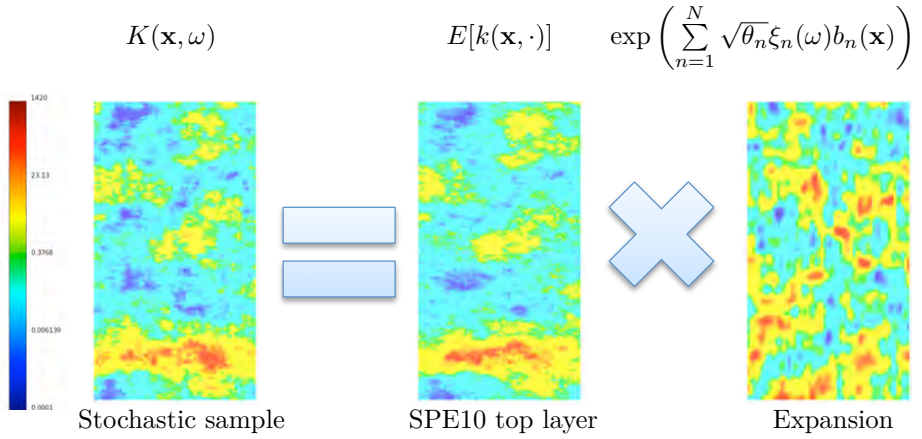


Figure 9.1: Truncated Karhunen-Loève expansion

Note that $(\theta_n, b_n(\mathbf{x}))$ are the eigenpairs of the covariance function

$$C(\mathbf{x},\mathbf{y}) = \int_\Omega \exp\left(-\frac{||\mathbf{x}-\mathbf{y}||_p}{\lambda}\right) d\Omega, \tag{9.8}$$

where λ is the correlation length. The Truncated Karhunen-Loève expansion was based on 38 eigenmodes in both the x- and y-directions. The quantity

of interest is here set to be the water cut in the top right well. The goal is to estimate the mean and variance of the water cut given uncertainty in the permeability field. Sequential simulations were were carried out in parallel on 60 cores of the Sierra cluster from Lawrence Livermore National Laboratory. A variance target of 10^{-5} is used as a convergence criteria. The results are seen in Table 9.1.

Level	#DoFs	#samples	$E[Q_l]$	$\mathrm{Var}[Q_l - Q_{l+1}]$	Cost per realization
Level 0 (fine)	66280	180	0.1112	0.00042182	100.8
Level 1	7409	840	0.1132	0.00149328	12.5
Level 2	639	2820	0.1211	0.00109587	0.75

Table 9.1: Numerical results of Multilevel Monte Carlo. The cost is measured in seconds.

As it is evident from Table 9.1, many more simulations are needed on the coarse levels compared to the fine grid level. In this case, the total cost of Multilevel Monte Carlo (MLMC) compared to standard Monte Carlo is

- MLMC: $180 \times 100.8 + 840 \times 12.5 + 2820 \times 0.75 \approx \mathbf{8.5}$ hours
- Standard Monte Carlo: $1020 \times 100.8 \approx \mathbf{28.6}$ hours

The standard Monte Carlo needed 1020 realization of the model to convergence to the same variance target as MLMC. Even for such a small problem (60×220 cells), significant computational savings are obtained. The savings will be even more substantial for larger problems.

Note that generating samples based on the Truncated Karhunen-Loève expansion is not a scalable approach. The memory requirement will dramatically increase for larger problem sizes.

Chapter 10

Developments in commercial simulator

The author has been part of the Industrial PhD programme in Denmark. This means that Lloyd's Register Consulting has been co-sponsoring and co-supervising the PhD. As a consequence, in addition to the academic research carried out over the course of the PhD, a significant amount of activities with a commercial purpose has also been performed. This chapter serves to document these activities.

Lloyd's Register Consulting is participating in a joint industry project, where further improvements to the reservoir simulator COSI is the primary focus. COSI (COmpositional reservoir SImulator) is a legacy oil and gas reservoir simulator. As the name reveals, it supports compositional simulations, however it also supports black oil simulations. The software was originally developed by the Danish National Laboratory, the Technical University of Denmark and COWIconsult. The legacy code has the following features

- 3D lowest order Finite Volume code with support for structured, cylindrical and unstructured grids. Standard upwinding scheme is used.
- Sequential Fortran 77 code using only one array for doubles and one array of integers.

- Three phases with water being immiscible. Any number of components.
- Isothermal simulator.
- Dual porosity/dual permeability.
- Fully implicit formulation based on backward Euler and Newton's method.
- Linear solver is TFQMR with ILU preconditioning.
- Peng-Robinson and Redlich-Kwong equations of state (and two modifications).
- Coupled wells.
- More than 100,000 lines of code.

10.1 Model equations

The governing equations in COSI consists of $i = 1, \ldots, \mathrm{NC}$ component mass conservation equations

$$\underbrace{\frac{\partial}{\partial t}\left[\phi_m \sum_j \left(\rho S C^i\right)_{jm}\right]}_{\texttt{Accumulation}} + \underbrace{\nabla \cdot \left[\sum_j \left(\rho C^i \mathbf{u}\right)_{jm} - \phi_m \sum_j \left(\rho S \{D^i\} \nabla C^i\right)_{jm}\right]}_{\texttt{Flow in and out}} = \underbrace{q^i_m + q^i_{Sm}}_{\texttt{Source}} \tag{10.1}$$

where the Darcy velocity is given by

$$\mathbf{u}_{jm} = -\frac{\{K k_j\}_m}{\mu_{jm}} \left(\nabla P_{jm} + g \rho_{jm} \nabla H\right) \tag{10.2}$$

In addition to the component mass conservation equations, the following constraint on saturations is also part of the system of equations

$$\sum_j S_{jm} = 1 \tag{10.3}$$

The various sizes involved in the equations above are listed below.

j : phase	ϕ : porosity
i : component	ρ : density
m : rock medium	S : saturation
C : concentration	K : permeability
D : diffusion	k : relative permeability
μ : viscosity	P : pressure
g : gravity	H : height

Note that in the component mass conservation equations, one of the source terms is for the wells and the other source term is for the matrix-fracture exchange. Notice also that the flow term (with the divergence operator) contains the well-known Darcy velocity, but also a diffusion term.

10.2 Parallelization and code modernization

The COSI code was provided to the author as a sequential Fortran 77 Microsoft Visual Studio compatible code. The goal was to parallelize the code to enable execution on clusters. Below is a list with some initial tasks carried out to accomplish this goal.

- Conversion to linux compatible code.
- Conversion from F77 fixed form to F90 free form.
- Changing all common blocks to Fortran modules form.
- Interfacing Fortran with C/C++.
- Change the static Fortran memory allocation to dynamic memory allocation in C/C++.
- Substituting the one real and one integer array to one array per variable.

With these changes in place, the code was prepared for parallelization using the PETSc library, [19, 18, 17]. PETSc is a general purpose software, which can be used for parallel solution of scientific applications modeled by partial differential equations. PETSc has many applications and can be used in various

ways. The first choice to make is whether to let PETSc handle the mesh encoding, mesh distribution and allocation of appropriate size vectors/matrices. We decided to let PETSc handle this part, since the implementation of an efficient distributed parallel communication layer is non-trivial. At the time we started this effort, PETSc had only recently been expanded with the DMPLEX mesh management feature, which enabled encoding of unstructured meshes. Since the inner routines of the COSI code was written in a way consistent with an unstructured mesh, we decided to use DMPLEX for mesh management despite the fact that it was a new addition to PETSc. As a consequence, we experienced some challenges along the way, but these were quickly resolved with the help of the PETSc development team (especially Matthew Knepley was a great help).

The DMPLEX mesh management framework was implemented in COSI by allocating all vectors/matrices to the appropriate local (to the parallel subdomain) size. Specifically, the vectors are the size of the number of cells in the subdomain plus the number of ghost cells. The for-loops in COSI were preserved but the range of values was shrinked from the global domain size to the local subdomain size plus the number of ghost cells. Based on the DMPLEX mesh encoding, the arrays holding the inner mesh relations in COSI were set up to match. In this way, most of the COSI code could be used in its original form.

DMPLEX is very general in its construction. It enables encoding an unstructured mesh expressed as a Hasse diagram. In our application, we require the ability to encode nonconforming meshes, where in parallel we need a ghost layer with an overlap of one. Figure 10.1 illustrates the type of mesh we need to be able to encode. Faces only partially overlap each other. In practice, each face may in fact overlap with 10 or more faces from other cells. This is typically a consequence of meshing a fault, where there is a displacement of part of the layer of porous media.

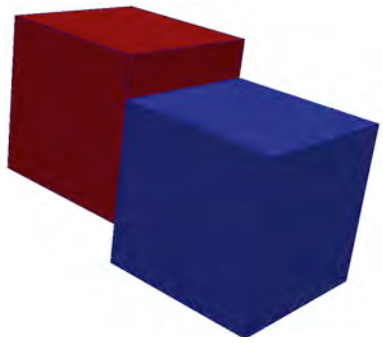


Figure 10.1: Illustration of the type of nonconforming mesh needed in COSI.

In COSI, we encode these types of meshes in DMPLEX in the following way. The following example is included, since we have not seen anywhere else a similar usage of DMPLEX and it may therefore be useful for others. The code below encodes two cubic cells sharing one face and four vertices (not the same as Figure 10.1). The two cones c0 and c1 represent the two cells. They are each defined by six faces, where face 7 is shared. The remaining cones: c2-c12 represent the 11 faces. They are each defined by four vertices. Note the unique numbering from 0-24. The last paragraph where DMPlexCreateLabel occurs is responsible for letting DMPLEX know that we ignore edges in our encoding.

```
DM dm;
PetscInt       dim = 3;
PetscInt       c0[6] =  {2,3,6,7,9,11};
PetscInt       c1[6] =  {4,5,7,8,10,12};
PetscInt       c2[4] =  {13,15,19,21};
PetscInt       c3[4] =  {14,16,20,22};
PetscInt       c4[4] =  {15,17,21,23};
PetscInt       c5[4] =  {16,18,22,24};
PetscInt       c6[4] =  {13,14,19,20};
PetscInt       c7[4] =  {15,16,21,22};
PetscInt       c8[4] =  {17,18,23,24};
PetscInt       c9[4] =  {13,14,15,16};
PetscInt       c10[4] = {15,16,17,18};
PetscInt       c11[4] = {19,20,21,22};
PetscInt       c12[4] = {21,22,23,24};
PetscMPIInt    rank;
PetscErrorCode ierr;

ierr = PetscInitialize(&argc, &argv, NULL, help);CHKERRQ(ierr);
ierr = DMCreate(PETSC_COMM_WORLD,&dm);CHKERRQ(ierr);
ierr = DMSetType(dm,DMPLEX);CHKERRQ(ierr);
ierr = DMSetDimension(dm, dim);CHKERRQ(ierr);
ierr = MPI_Comm_rank(PetscObjectComm((PetscObject) dm), &rank);CHKERRQ(ierr);

if (!rank) {
  ierr = DMPlexSetChart(dm,0,25);CHKERRQ(ierr);
  ierr = DMPlexSetConeSize(dm,0,6);CHKERRQ(ierr);
  ierr = DMPlexSetConeSize(dm,1,6);CHKERRQ(ierr);
  ierr = DMPlexSetConeSize(dm,2,4);CHKERRQ(ierr);
  ierr = DMPlexSetConeSize(dm,3,4);CHKERRQ(ierr);
  ierr = DMPlexSetConeSize(dm,4,4);CHKERRQ(ierr);
  ierr = DMPlexSetConeSize(dm,5,4);CHKERRQ(ierr);
  ierr = DMPlexSetConeSize(dm,6,4);CHKERRQ(ierr);
  ierr = DMPlexSetConeSize(dm,7,4);CHKERRQ(ierr);
  ierr = DMPlexSetConeSize(dm,8,4);CHKERRQ(ierr);
  ierr = DMPlexSetConeSize(dm,9,4);CHKERRQ(ierr);
  ierr = DMPlexSetConeSize(dm,10,4);CHKERRQ(ierr);
  ierr = DMPlexSetConeSize(dm,11,4);CHKERRQ(ierr);
  ierr = DMPlexSetConeSize(dm,12,4);CHKERRQ(ierr);
}
ierr = DMSetUp(dm);CHKERRQ(ierr);
```

```
if (!rank) {
  ierr = DMPlexSetCone(dm,0,c0);CHKERRQ(ierr);
  ierr = DMPlexSetCone(dm,1,c1);CHKERRQ(ierr);
  ierr = DMPlexSetCone(dm,2,c2);CHKERRQ(ierr);
  ierr = DMPlexSetCone(dm,3,c3);CHKERRQ(ierr);
  ierr = DMPlexSetCone(dm,4,c4);CHKERRQ(ierr);
  ierr = DMPlexSetCone(dm,5,c5);CHKERRQ(ierr);
  ierr = DMPlexSetCone(dm,6,c6);CHKERRQ(ierr);
  ierr = DMPlexSetCone(dm,7,c7);CHKERRQ(ierr);
  ierr = DMPlexSetCone(dm,8,c8);CHKERRQ(ierr);
  ierr = DMPlexSetCone(dm,9,c9);CHKERRQ(ierr);
  ierr = DMPlexSetCone(dm,10,c10);CHKERRQ(ierr);
  ierr = DMPlexSetCone(dm,11,c11);CHKERRQ(ierr);
  ierr = DMPlexSetCone(dm,12,c12);CHKERRQ(ierr);
}
ierr = DMPlexSymmetrize(dm);CHKERRQ(ierr);

if (!rank) {
  DMLabel  label;
  PetscInt i;
  ierr = DMPlexCreateLabel(dm, "depth");CHKERRQ(ierr);
  ierr = DMPlexGetDepthLabel(dm, &label);CHKERRQ(ierr);
  for (i = 0; i<25; ++i) {
    if (i<2)      {ierr = DMLabelSetValue(label, i, 3);CHKERRQ(ierr);}
    else if(i<13) {ierr = DMLabelSetValue(label, i, 2);CHKERRQ(ierr);}
    else          {
      if (i==13) {ierr = DMLabelAddStratum(label, 1);CHKERRQ(ierr);}
      ierr = DMLabelSetValue(label, i, 0);CHKERRQ(ierr);
    }
  }
}
```

The above code only encodes the mesh relations. In addition to this, we fill out a number of vectors with the various information needed for the discretization and 3D outputting, e.g. face area, vertex coordinates, etc. Since PETSc did not already contain this, we also wrote a parallel 3D outputter for arbitrary polyhedrons based on Paraview's VTU file format (type 42).

In addition to using PETSc's DMPLEX, we also chose to use the KSP interface. PETSc provides the following three interfaces

- TS - Time stepping interface for the solution of ODEs and DAEs. In the context of solving the fully implicit reservoir simulation equations, you would need to provide functions to evaluate e.g. the residual and the Jacobian and the TS framework will take care of the rest. TS has a pointer to a nonlinear solver (SNES).
- SNES - Nonlinear solver interface. Provide functions to evaluate e.g. the

residual and the Jacobian and the SNES framework handles the rest. SNES has a pointer to a linear solver (KSP).

- KSP - Linear solver interface. Provide matrix and right hand side and KSP can handle the rest.

In the opinion of the author, it is best to choose the TS interface if possible, however the COSI code contains a lot of logic regarding well controls, error outputs, truncation error estimates, step size controllers, etc, which unfortunately is bundled together with the time stepper and the nonlinear solver making it difficult to straightforwardly use TS or SNES. As a consequence, the KSP interface was chosen as a start. A further development to use the TS interface is already in progress.

One of the more challenging parts of the parallelization process was the well model. Since the well model couples non-neighbour cells together, the normal ghost layer between subdomains is insufficient. This was resolved by introducing an MPI communicator per well and with calls to MPI_Reduce, the values necessary for the well model to function were communicated to all processors, which contained part of the well. It was done by retrofitting existing COSI arrays that described the relations between reservoir cells and the cells containing wells.

10.3 New parallel linear solver

The original COSI software used a sequential TFQMR solver with ILU preconditioning. For the new and parallel version of COSI, we implemented a parallel preconditioner based on the PETSc framework. The choice of preconditioner in COSI is a derivative of the standard solver in state-of-the-art commercial and research simulators. The preconditioner is the so-called Constrained Pressure Residual method (CPR), [117]. The method works by first targeting the low frequency errors from the pressure part of the equations and then resolve the remaining high frequency errors. This is accomplished by restricting the full system into a scalar system for the pressure in a gaussian elimination type procedure. This scalar system is solved (typically with a multigrid solver), which results in a correction to the pressure equation in the full system. The purpose of this correction is to remove low frequency errors. The corrected full system is then preconditioned with another method to remove the remaining high frequency errors.

The CPR method can be formulated as:

$$M_{cpr}^{-1} = M^{-1}(I - \hat{A}CA_p^{-1}C^T) + CA_p^{-1}C^T, \tag{10.4}$$

where M is some preconditioner and

$$\hat{A} = WA. \tag{10.5}$$

Here W is some matrix performing the gaussian elimination type operation on A. The scalar pressure matrix is given by

$$A_p = C^T \hat{A} C. \tag{10.6}$$

C is given by

$$C = \begin{bmatrix} e_p & & & \\ & e_p & & \\ & & \ddots & \\ & & & e_p \end{bmatrix}, \tag{10.7}$$

where

$$e_p = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{10.8}$$

Here e_p is as long as the number of unknowns per cell. The first entry of e_p is 1 if the pressure equation is ordered the first in the system. This assumes the unknowns are ordered in a cell-based manner. With these matrices introduced, the CPR preconditioner method follows:

1. Restrict full system residual: $r_p = C^T W r$.
2. Solve $A_p x_p = r_p$.
3. Correct the full system residual: $r_c = r - ACx_p$.
4. Solve: $Mx = r_c$
5. Return the preconditioned vector y, which is the sum of full system and pressure solutions: $y = x + s$.

In state-of-the-art preconditioners for reservoir simulation, it is common to use algebraic multigrid (AMG) to solve the pressure system and parallel ILU to solve the corrected full system. In this work, we found that parallel ILU quickly became a bottleneck due to poor scalability. As a consequence, the parallel ILU for preconditioning the corrected full system was switched out with restricted

additive overlapping schwarz. An overlap of 1 was used and for each subdomain, a sequential ILU(0) implementation was applied. By experimentation it was found that restricted additive overlapping schwarz provided much better scalability, while still proving to be an efficient method for removing high frequency errors.

The preconditioner was implemented using the PETSc PCSHELL framework. Here three functions are needed, namely a

- Create() - Typically responsible for preallocation of memory. This functions is only called once in COSI.
- Setup() - This function is called every time the Jacobian matrix changes. In the context of COSI, this is for every Newton iteration.
- Apply() - This function is called for each linear solver iteration. This is where the preconditioner is applied.

The preconditioner was accelerated by GMRES. For the first stage solve (pressure part) of CPR, BoomerAMG from Hypre was used. For the second stage (corrected full system solve), standard PETSc components were used (ASM with overlap 1 + ILU(0) subsolves).

10.4 Scaling study

A scaling study of the new parallel COSI is presented in the following. The basis of the test is the SPE1 case. The model is modified to allow for easily changing the number of cells. Furthermore, to allow the test to run in a reasonable time frame, a shorter time horizon is used compared to the original SPE1. The results are plotted in the following graphs. The scalability of the following components of the parallel COSI code are analyzed. For each component of the code, comments are made to address the results in the plots below. Figure 10.2 provides some basic information for the simulations carried out. These include the average newton iterations per time step, the number of time steps and the average number of linear iterations per newton iteration.

- **GLOBAL TIMER (Figure 10.3):** This measures the full execution time from start to end. The results indicate scalability to a point around 70-80 thousand cells per processor in terms of weak scaling and 64 cores in terms of strong scaling. The weak scaling is given by looking at the plot

on the bottom right of Figure 10.3. Specifically, optimal weak scaling is a horizontally straight line. The strong scaling can be read from the top right plot in Figure 10.3. Note that these results are heavily affected by the fact that the transient phase of the code is relatively small compared to the setup phase.

- **Setup (Figure 10.4):** Loading input files, setup and distribution of mesh. The setup is inherently sequential and therefore it does not scale across processors. Since these results have been produced, the setup times have been improved significantly due to improvements in PETSc and due to a new implementation for the encoding of the mesh in DMPLEX.

- **Transient (Figure 10.5):** Starts after the setup phase and completes at the end. This part of the code is fully parallel. It includes everything except the setup. As expected, the results show better strong scalability compared to the GLOBAL TIMER (up to 256 cores), but the weak scaling is still around the same as for the GLOBAL TIMER. This is largely due to the fact that the solvers do not scale perfectly with problem size as it is evident from Figure 10.2.

- **Linear solver (Figure 10.6):** GMRES+CPR(AMG+Restricted Additive Overlapping Schwarz(ILU(0))). Note that this is plotted as average time per newton iteration. The strong scaling is very similar to that of the transient timer, however the weak scaling is improved to between 32-65 thousand cells per processor. This is probably due to the fact that the slight increase in newton iterations is discarded when plotting this as an average.

- **CPR AMG (Figure 10.7):** Algebraic multigrid: BoomerAMG. Note that this is plotted as an average time per linear iteration. The strong scaling is similar to that of the linear solver. Clearly the AMG component of the linear solver dictates scalability of the overall linear solver. The weak scaling is slightly improved. This is probably due to the fact that the increase in linear iterations is discarded when plotting this as an average.

- **CPR ILU factorize (Figure 10.8):** ILU(0) factorization. The factorizations occur on each subdomain and it is therefore inherently parallel. The results also indicate that this component scales well.

- **CPR ILU solve (Figure 10.9):** Application of Restricted Additive Overlapping Schwarz after ILU factorization. This component scales well.

- **Assembly (Figure 10.10):** Assembly of Jacobian in Compressed Sparse Row format. The strong scaling of the assembly is good but not perfect for small problem sizes. The weak scaling is good. The assembly could be improved in the current implementation by supplying PETSc with more

values at a time, however the assembly is far from being the bottleneck of the code at the moment.

- **FUNCPV (Figure 10.11):** The name refers to a function in COSI which handles all the physics of the simulator. Since this part of the code is embarrassingly parallel, good scaling is expected. The results also indicate almost perfect scalability.

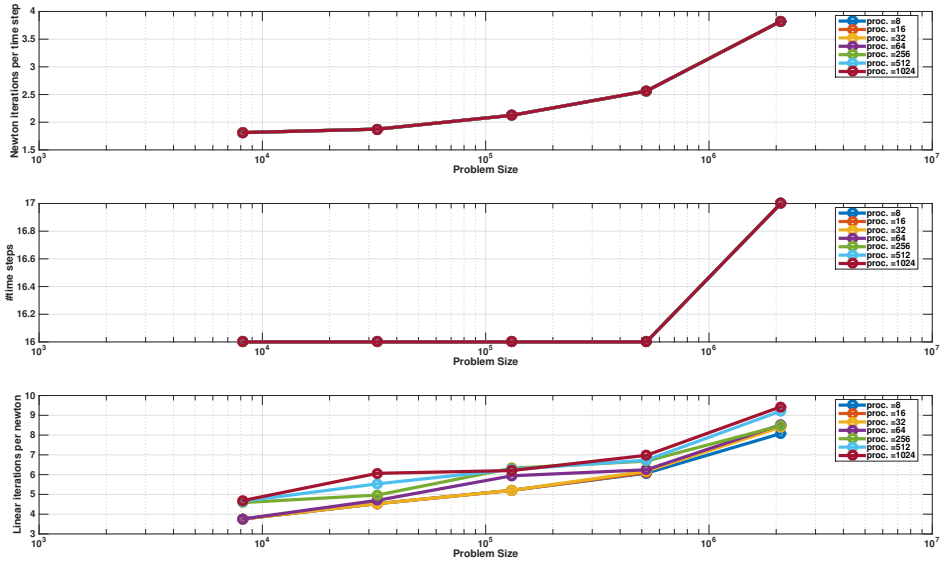


Figure 10.2: Basic information for simulations

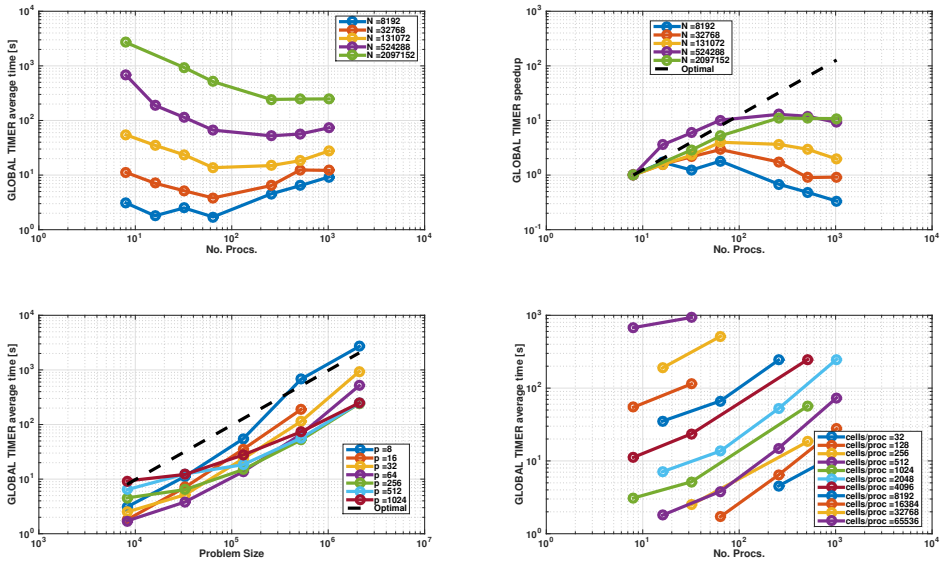


Figure 10.3: Global timer

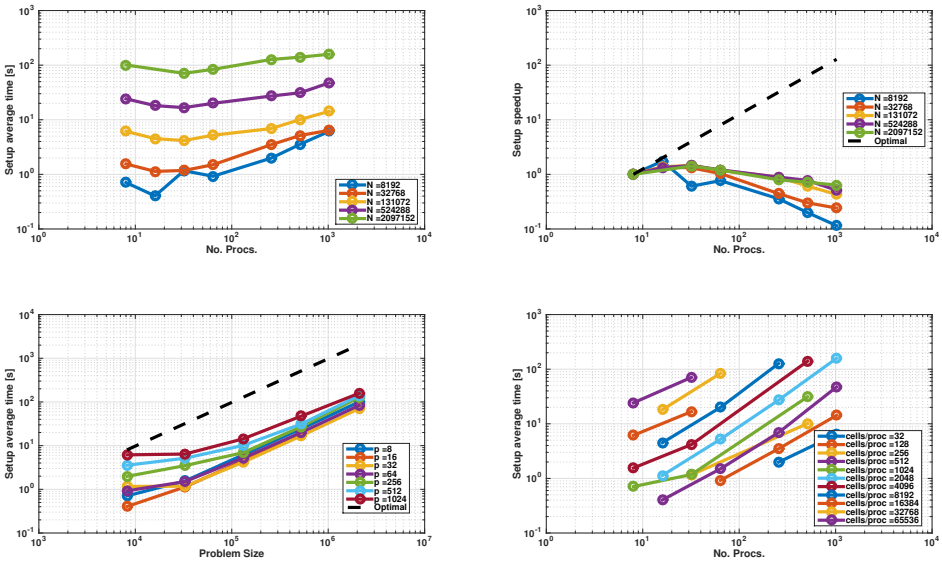


Figure 10.4: Setup

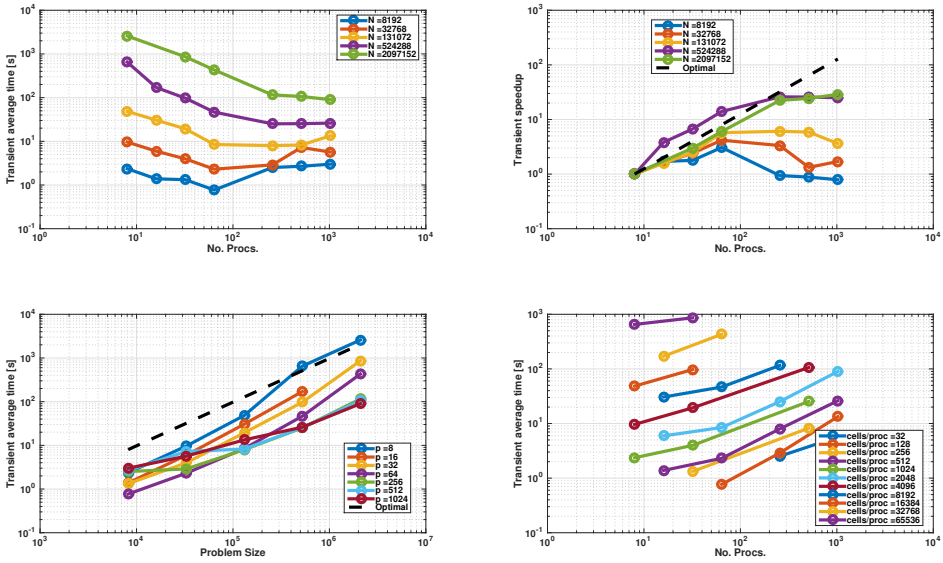


Figure 10.5: Transient

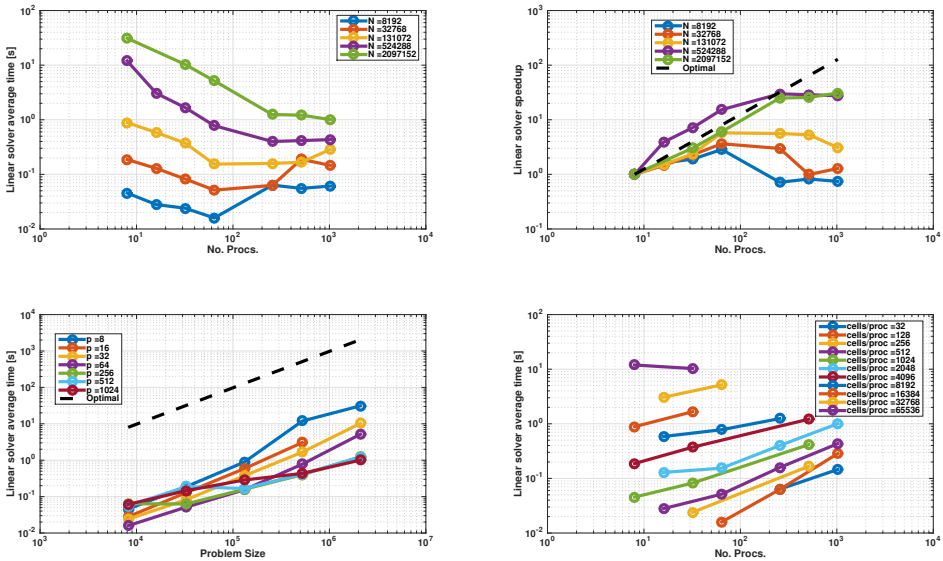


Figure 10.6: Linear solver

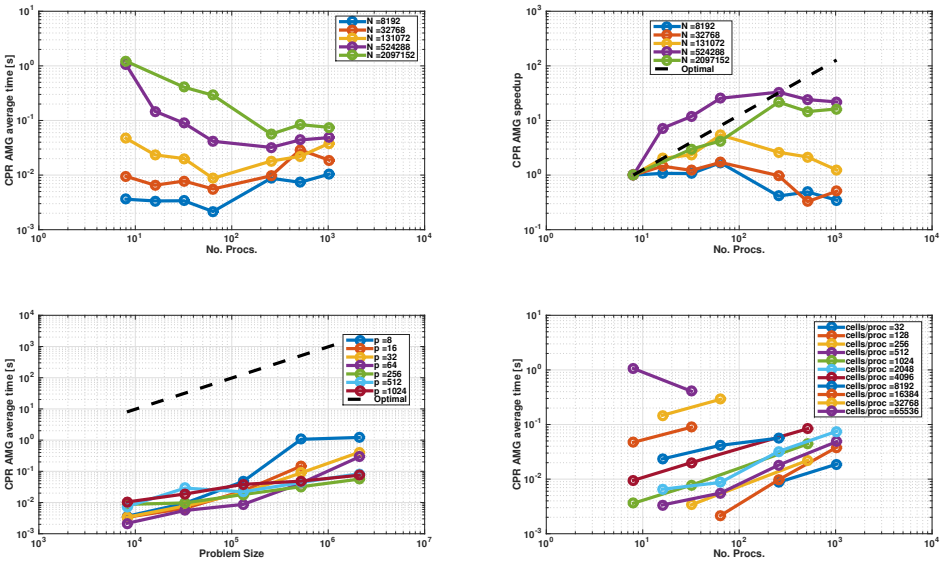


Figure 10.7: CPR AMG

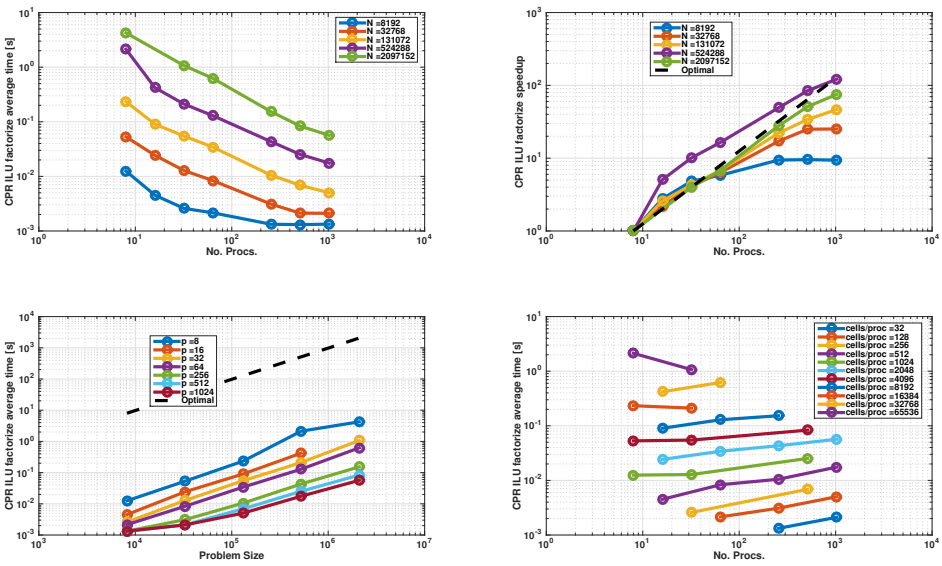


Figure 10.8: CPR ILU factorize

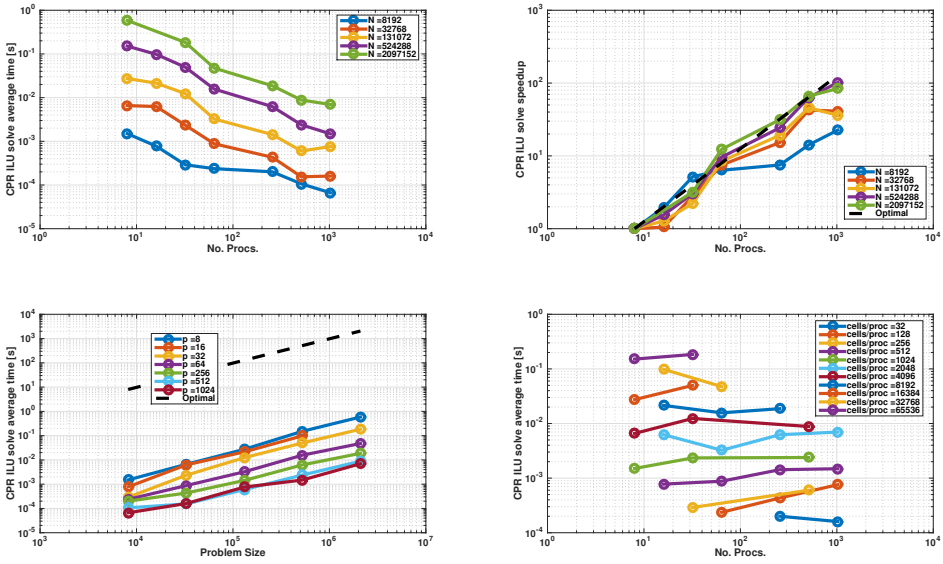


Figure 10.9: CPR ILU solve

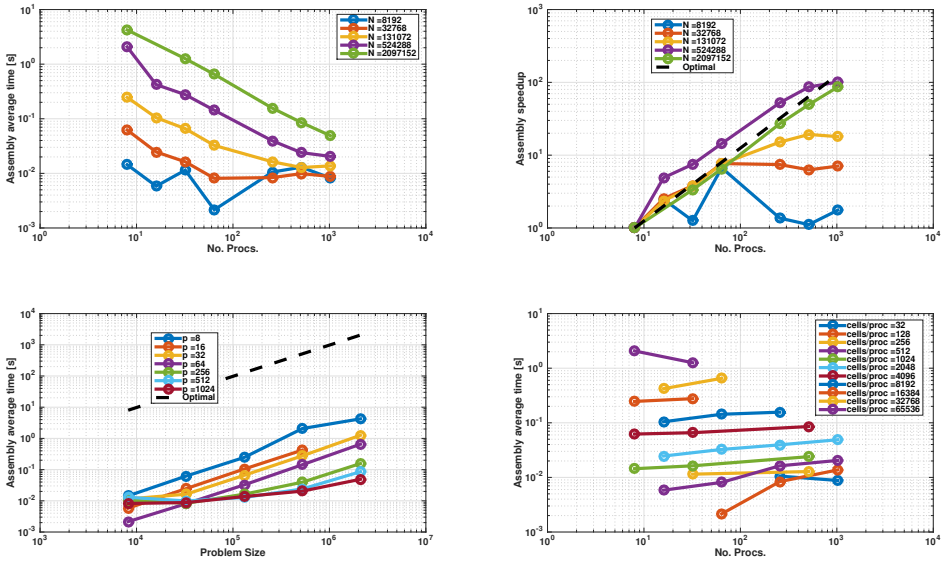


Figure 10.10: Assembly

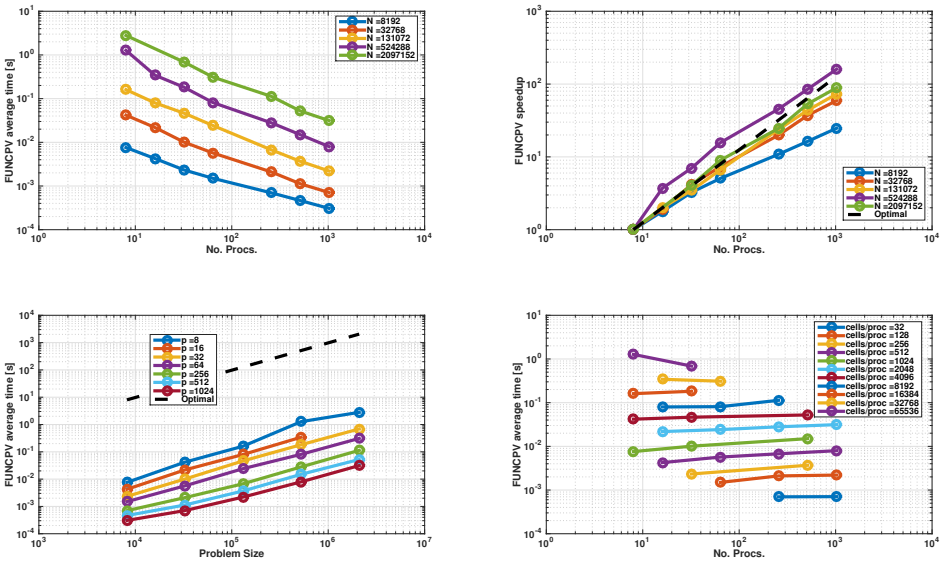


Figure 10.11: FUNCPV (physics)

10.5 Polynomial preconditioner based on Neumann series

This section will compare a polynomial preconditioner to the CPR(AMG+RAS(ILU)) preconditioner introduced previously in this chapter. The motivation for studying a polynomial preconditioner is the inherent parallelism of the algorithm. The goal is to improve strong scaling compared to the CPR(AMG+RAS(ILU)) preconditioner. The CPR(AMG+RAS(ILU)) preconditioner is based on algebraic multigrid (AMG), which is great for weak scaling purposes but suffers in terms of strong scaling, when the number of degrees of freedom per core becomes too small. The polynomial preconditioner is based on a Neumann series. The specific preconditioner is explained in [45]. In this work, the preconditioner is implemented in PETSc using the PCSHELL concept.

The basis of the polynomial preconditioner considered in this work, is the splitting of the system matrix A into two matrices P and E such that

$$A = P + E. \tag{10.9}$$

Given this splitting, we can write

$$A = P + E = (I + EP^{-1})P, \tag{10.10}$$

Letting P contain the most dominant terms of A, while still being relative easy to invert, a polynomial preconditioner based on Neumann series can be formed:

$$M^{-1} = \left[I + \sum_{k=1}^{N} (-1)^k (P^{-1}E)^k \right] P^{-1}. \tag{10.11}$$

The components of the resulting preconditioner consist of an (approximate or exact) solve using the matrix P and a sparse matrix-vector product with the matrix E. Algorithm 5 provides a pseudo-code for the implementation.

Algorithm 5 Pseudo code for preconditioning by Neumann series.
Input:
Input vector: x
Polynomial order: N
Output:
Output vector y

Solve $Pu^0 = x$ to find u^0.
Set $y = u^0$
for $k = 1 \dots N$ **do**
    $w^k = Eu^{k-1}$
    Solve $Pu^k = w^k$ to find u^k.
    $y = y + (-1)^k u^k$.
end for

Performance for different polynomial orders have been investigated. In general $N = 3$ has proven to be a good compromise between computational efficiency of the code and convergence rate. By experimentation it was found that the optimal choice of N is case-dependent.

Discussion on which terms to include in P

The method relies on extracting the most important non-zeros of A and including them in P, where they are used in the "inversion" (actually a solve). The important non-zeros of A are thought to be those with the strongest couplings in the system. In this implementation, the strongest couplings are identified in

a setup stage based on a quantity called the transimissiblity:

$$T_{ij} = \frac{a_{ij} K_{ij}}{h_{ij}}, \tag{10.12}$$

where a is the face area, K is the permeability and h is the distance. The subscript ij denotes that it is between two neighbour cells: i and j. The transmissibility roughly dictates the amount of flow between two cells and therefore it can be considered a good static quantity to determine strong couplings in the system.

In this work, two different methods have been implemented for constructing the matrix P. They are described in the following. In both methods, P is constructed such that it contains no processor off-diagonal nonzeros (parallel CSR format terminology). This means that no communication is needed for computations involving P. Essentially the global P can be considered as a collection of local on-processor P matrices. The choice of keeping all communications in E rather than in P is motivated by the fact that a solve involving the P matrix is significantly cheaper, when it is local to the processor.

P as a block tridiagonal matrix

In the first implementation, P was constructed by tracing maximum transmissiblity lines in the mesh and extracting the entries from A corresponding to the cells forming the lines. Since, each cell in a line can have a maximum of two neighbours, the resulting P matrix becomes block tridiagonal. The goal of tracing lines in the mesh was to extract the strong couplings in the system and at the same time arrive at an easily invertible system P. Figure 10.12 displays the system matrix (here called J), E and P.

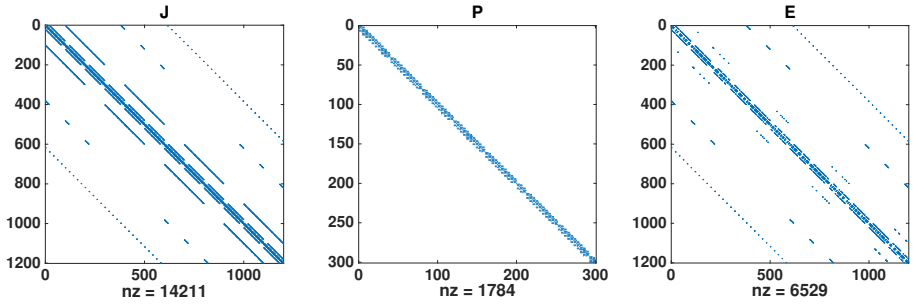


Figure 10.12: Sparsity pattern of matrices using 4 processors.

P constructed based on a list of largest transmissibilities

In the second implementation, P was constructed more generally by simply including a percentage of the largest transmissibility couplings within each subdomain (representing non-zeros in A) in addition to the diagonal of A. This was found to be easier to implement while providing a potential for strengthening the convergence. Of course, this approach may result in some block rows of P to contain many nonzeros and other block rows of P to have only the block diagonal entries. Different percentages of largest transmissibility couplings was tested for and a percentage value of 20-40% was found to provide a working preconditioner. Figure 10.13 displays the system matrix (here called J), E and P. Notice that operations involving P still requires no communication.

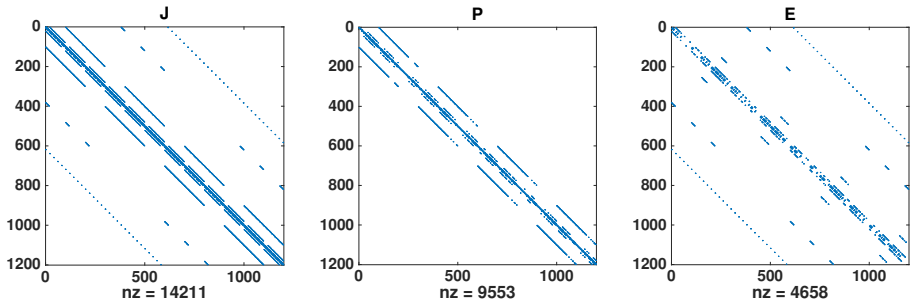


Figure 10.13: Sparsity pattern of matrices using 4 processors.

"Inverting" P

Two different methods have been applied, namely LU and ILU. Both were applied to P on each subdomain. By experimentation it was found that LU factorization works well for small subdomains but became too expensive when the size of the subdomains is increased. To remedy this issue, ILU(0) factorization was used instead.

10.5.1 Numerical results

In both the CPR(AMG+RAS(ILU)) linear solver used for comparison and the polynomial preconditioner introduced here, the CPR methodology is applied. That is, the preconditioner is split into two stages and both the pressure solve

and the full system solve is done with the polynomial preconditioner. Simulations were run using a standard benchmark case, namely the SPE10 model 2, [95]. The test case models the injection of water and resulting displacement of oil in the subsurface. It is a two-phase nearly incompressible model. The grid is 60x220x85 grid cells. The numerical difficulty of the problem arises from the highly heterogeneous permeability coefficient displayed in Figure 10.14.

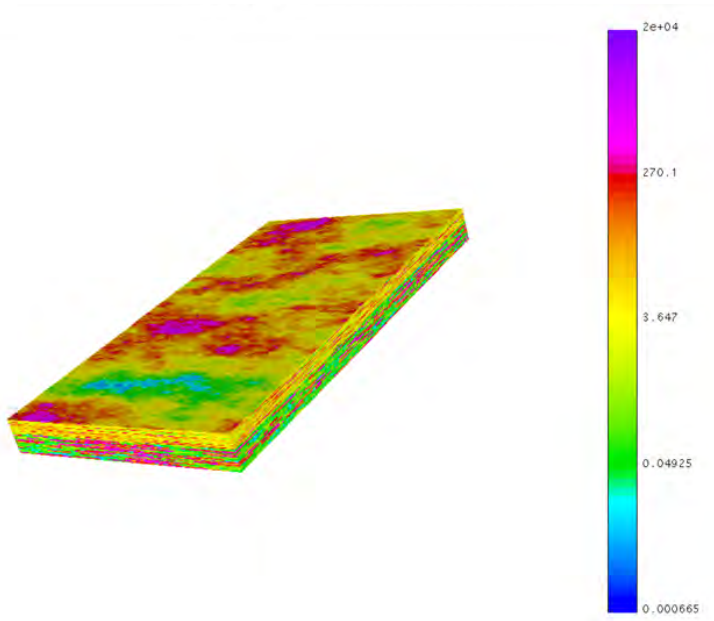


Figure 10.14: SPE10 permeability field

In the following, only results the first method of constructing P (block tridiagonal) will be studied. Method 2 was by experimentation found to be slightly less performant than method 1. In the following we will be comparing:

- CPR with AMG for the pressure solve and restricted additive Schwarz (RAS) with ILU(0) subsolves for the corrected full system.
- CPR with Neumann preconditioning for both the pressure solve and the corrected full system. The polynomial order is set to $N = 3$ and the length of lines is maximum 10 grid cells.

The study will focus on the hypothesis that the polynomial preconditioner can achieve better strong scaling and therefore eventually outperform the CPR(AMG+RAS(ILU))

as the number of cores increases. Figure 10.15 displays the results obtained from running on the Computerome cluster from DTU Risoe. For the linear solver, a relative linear tolerance of 10^{-4} and absolute tolerance of 10^{-6} was used. 10 time steps with constant time step sizes were simulated. The strong scaling is clearly better for the polynomial preconditioner, however the convergence rate of the CPR(AMG+RAS(ILU)) solver is superior and therefore it remains the better choice for this case.

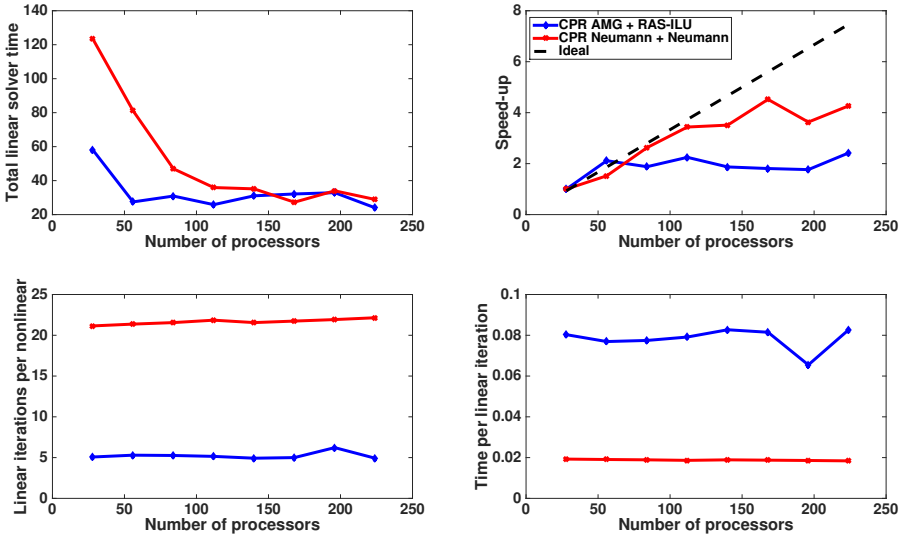


Figure 10.15: Strong scaling

If the CPR Neumann Neumann implementation can be improved to achieve better convergence rate, it could potentially outperform the CPR(AMG+RAS(ILU)) solver for high core-counts. The convergence rate of the method relies heavily on the distribution of nonzeros between P and E. In this work, two different approaches has been tested but none of them are able to obtain good enough convergence rate to compete with the CPR(AMG+RAS(ILU)) solver. The computations were carried out on the Computerome cluster. Each node is equipped with the following hardware:

- 2x Intel Xeon Processor E5-2683 v3 (14-core, 2.00GHz)
- 128 GB memory

- Mellanox SX6036 36-port 56Gb/s InfiniBand

Chapter 11

Conclusions

A number of numerical multilevel techniques have been implemented and studied for various porous media flow models. This section will attempt to summarize the key findings in each study. In this work, different multigrid solvers and preconditioners for linear systems of equations have been implemented and studied. For mixed systems, multigrid preconditioners based on AMGe has been compared to state-of-the-art block diagonal preconditioners based on either ADS or AMG. It was found that while the multigrid preconditioners based on AMGe typically improved convergence rates compared to the block diagonal preconditioners, the heavy computational cost associated with the construction of the AMGe coarse spaces results in the block diagonal preconditioners to outperform those based on AMGe. However, the picture changes completely in the case of solving variationally upscaled problems. Due to the AMGe upscaling technique producing upscaled systems with different spectral properties than those for fine grid systems, the standard block diagonal preconditioners are very inefficient for these systems. However, the AMGe based multigrid preconditioners are largely unaffected by the change in spectral properties and consequently they outperform the block diagonal preconditioners for the upscaled systems. Furthermore, there is a huge potential in the reuse of the AMGe coarse spaces for nonlinear time-dependent PDEs. With sufficient reuse of the coarse spaces, it may be possible for AMGe based multigrid solvers to outperform the block diagonal preconditioners even for fine grid systems.

In Paper I, FAS with a z-line Gauss-Seidel smoother was compared to Newton's method with a FGMRES+CPR(AMG+ILU(0)) linear solver for a 3D immiscible black oil formulation with a multi-cell well model. Furthermore a hybrid FAS/Newton solver was implemented and tested. It was found that for the given model equations and range of problems considered, FAS or hybrid FAS/Newton outperformed the traditional Newton's method in terms of convergence rates, computation time, memory requirements and robustness.

In Paper II, a novel AMGe technique with improved approximation properties was used to implement a variational upscaling tool based on an improved IMPES formulation of the mixed incompressible fluids/rock reservoir simulation equations. It was found that the technique provided a flexible way to construct a multilevel hierarchy of variationally upscaled models with a high accuracy. Multilevel results show that the errors as a function of coarsening increase at a sublinear rate and therefore demonstrate that the AMGe coarse spaces lead to more accurate results than solving the same problem using standard finite elements on an equally coarse grid (i.e. for the same number of degree of freedom). Furthermore, it was found that even though the numerically upscaled models were accurate, some numerical diffusivity should be expected. This typically resulted in the very coarse upscaled models to underestimate the exact time when water breakthrough occurs. Finally, a number of agglomeration techniques (grouping of the fine grid elements into agglomerates to get a coarse mesh) were studied. It was found that the choice of agglomeration had a large impact on the quality of the upscaled models. It is important to leave the elements containing wells and (possibly) their immediate neighbors unagglomerated to capture the near-well flow accurately. Furthermore, due to the strong coupling in the vertical direction, only agglomerating in the x- and y-directions gives a significantly better upscaled approximation.

In Paper III, it was demonstrated how the same hierarchy of AMGe coarse spaces could be used both for variational upscaling purposes and as a multigrid solver for the upscaled systems. Furthermore, the AMGe upscaling proved to be able to approximate a fine grid reference solution with better accuracy than traditional flow-based upscaling methods (averaging of coefficients and rediscretization on a coarser mesh). Finally, we demonstrated that a parallel implementation of our IMPES-based reservoir simulator using state-of-the-art algebraic multigrid solvers shows good strong scaling behavior up to hundreds of MPI processes for the SPE10 problem.

In Paper IV, we circled back to FAS and combined it with the AMGe technique to obtain a nonlinear multigrid solver for unstructured meshes and with highly accurate interpolation operators. The FAS-AMGe solver was tested on a challenging saddle point problem. The solver was compared to Newton's method and Picard iterations (both exact and inexact versions). FAS outperformed

the exact methods (4X faster) and also proved slightly faster than the inexact versions.

To demonstrate one application of AMGe variationally upscaled models, a simple study of Multilevel Monte Carlo (MLMC) simulations was carried out. In particular, the software implemented for Paper II was used as a forward model in the MLMC simulations. The permeability field was considered as a stochastic variable and truncated Truncated Karhunen-Loève expansions were used to generate stochastic samples. The numerical experiments were based on the top layer of the SPE10 model and the MLMC simulations were used to estimate the mean and variance for the water cut in a producer. Using 60 processors of a cluster, MLMC was compared to standard Monte Carlo simulations. It was found that MLMC converged in 8.5 hours, whereas standard Monte Carlo simulations needed 28.6 hours.

Finally, the distributed parallelization (using PETSc) of the in-house compositional reservoir simulator COSI is described. Parallel scaling has been studied up to a thousand processors and each component of the code has been analyzed separately. Decent strong and weak scaling was observed. A polynomial preconditioner was investigated as an alternative to the current solver scheme. The goal was to obtain a preconditioner with improved strong scaling. Experiments confirmed this to be true, however the results also showed that the solver based on GMRES with CPR preconditioning using Algebraic Multigrid and Restricted Additive Overlapping Schwarz with ILU(0) subsolves proved to be the strongest despite the better strong scaling behavior for the polynomial preconditioner. This was largely due to the fact that the CPR(AMG+RAS(ILU(0))) preconditioner resulted in significantly better convergence rates compared to the polynomial preconditioner.

Chapter 12

Perspectives

This chapter contains thoughts and ideas for further improvements to the topics and techniques studied in this thesis.

Perhaps the largest obstacle in the application of AMGe to the reservoir simulation equations is the inherent reliance on the finite element method. Historically, finite volume schemes are by far the most common for porous media flow. However, in the opinion of the author, it seems there is a general trend towards discretization of fluid flow problems using finite elements rather than finite volume schemes (discontinuous galerkin finite element methods also included). This may not be the case in industry yet, but research continues pushing frontiers. This general trend may be a consequence of finite elements more easily allowing higher order discretizations on unstructured meshes (beyond second order) and the fact that the finite element method is rich in theory, in turn providing the user with more tools, which can be used for various purposes. AMGe is a perfect example of this. Here finite element basis functions (or stiffness matrices) can be used for constructing methods for variational upscaling, linear solvers and nonlinear solvers. Also the application of finite element based schemes may help with obtaining a general higher-order discretization method and reduce grid orientation effects. As a consequence of pursuing finite element based techniques, some of the nice things that industry have gotten used to with lowest-order finite volume schemes, now requires extra attention. For instance, most reservoir fields are currently meshed with a nonconforming mesh or otherwise known

as a pillar grid (or corner-point grid). This type of mesh serves to minimize the number of cells required to satisfactorily mesh a layered structure such as porous media. With noncorforming meshes, faults and fractures are more easily treated. The problem is that many finite element methods rely on a conforming mesh. To overcome this issue, either we remesh the reservoir fields in a conforming way, or we investigate finite element discretization methods capable of meshing nonconforming meshes. The first suggestion would result in more cells/elements and put some restriction on the flexibility in the meshing. With modern hardware, the extra elements should not be a big deal, however limiting the flexibility for the mesher is, in the opinion of the author, not great. The latter suggestion has already seen a great deal of interest. The extended finite element method and ideas from mortar elements seem like good candidates.

A natural extension of the work in this thesis (in particular Paper IV), is the implementation of a fully implicit velocity/pressure/saturation formulation using AMGe for variational upscaling and as a nonlinear solver (for the upscaled problems) at the same time. In Paper IV, a nonlinear mixed problem for velocity and pressure is solved using FAS-AMGe. To extend this to a black oil simulator setting requires the introduction of the transport (saturation) equations. If a standard global linearization scheme were to be employed (e.g. Newton's method), the resulting Jacobian would be a saddle point problem with some non-standard terms for which we are not familiar with any efficient preconditioner. Instead, we see a potential in applying FAS-AMGe as a solver for these nonlinear systems. It should be noted that Paper IV relies on global linearization and as a smoother uses a particular AMGe based preconditioner (Multilevel Divergence Free) for that particular system, however in the opinion of the author, if FAS-AMGe is to be applied more generally, it should be done so by using local linearization and a less problem-specific smoother (e.g additive overlapping schwarz type).

Another interesting study could be an extension of Paper I to unstructured meshes. Paper I investigates the application of FAS to solve the standard finite volume formulated reservoir simulation equations on structured grids. Whereas AMGe based techniques still requires maturing to be used in industry simulators, a simpler agglomeration based finite volume FAS solver could more easily be implemented in current industry simulators. Paper I demonstrates that the potential of FAS to accelerate reservoir simulators is great. However, it remains to be seen if a version for unstructured meshes is robust enough to deal with industry cases. The biggest unknown here is if the restriction and prolongation would be accurate enough to provide an efficient multigrid solver. Also an appropriate parallel smoother is required. Here, ideas from parallel smoothers in algebraic multigrid could probably be used.

In this work, some experimentation was carried out with a spectral AMGe

method for the upscaling of the model in Paper II. This is however undocumented, since it was somewhat off-topic and required significant more effort to constitute a full study. One of the key motivators for applying the spectral AMGe method is for improving the accuracy of the L^2 (pressure and saturation) coarse spaces. In the version of AMGe used in this work, only one degree of freedom for each agglomerated element is allowed in the upscaled problem for the L^2 space. We have observed that the largest error (between finest grid and upscaled solution) typically is for the pressure. This could potentially be improved by applying a spectral AMGe method, where a variable number of degrees of freedom is used. The actual number of degrees of freedom depends on a tolerance set by the user. It should be noted that the spectral AMGe methods also have their downsides, since the construction of coarse spaces is done by solving local eigenvalue problems, which for large agglomerates may be very expensive.

Bibliography

[1] MFEM: Modular finite element methods. `mfem.org`.

[2] MFEM: Modular finite element methods. `mfem.org`.

[3] V. E. Henson J. E. Jones T. A. Manteuffel S. F. McCormick G. N. Miranda A. J. Cleary, R. D. Falgout and J. W. Ruge. Robustness and scalability of algebraic multigrid. *SIAM J. Sci. Comput.*

[4] M. G. Madsen A. P. Engsig-Karup and S. L. Glimberg. A massively parallel GPU-accelerated model for analysis of fully nonlinear free surface waves. *Int. J. Num. Meth. Fluids,* **70***:1, 2011.*

[5] J. Aarnes. On the use of a mixed multiscale finite element method for greater flexibility and increased speed or improved accuracy in reservoir simulation. *Multiscale Modeling; Simulation*, 2(3):421–439, 2004.

[6] J. Aarnes, S. Krogstad, and K. Lie. A hierarchical multiscale method for two-phase flow based upon mixed finite elements and nonuniform coarse grids. *Multiscale Modeling; Simulation*, 5(2):337–363, 2006.

[7] Jørg E Aarnes, Vegard Kippe, and Knut-Andreas Lie. Mixed multiscale finite elements and streamline methods for reservoir simulation of large geomodels. *Advances in Water Resources*, 28(3):257–271, 2005.

[8] Jørg E Aarnes, Stein Krogstad, and Knut-Andreas Lie. Multiscale mixed/mimetic methods on corner-point grids. *Computational Geosciences*, 12(3):297–315, 2008.

[9] J. R. Appleyard. Nested factorization. *SPE Reservoir Simulation Symposium, 15-18 November 1983, San Francisco, California.*

[10] Todd Arbogast. An overview of subgrid upscaling for elliptic problems in mixed form. *Contemporary Mathematics*, 329:21–32, 2003.

[11] Todd Arbogast. Analysis of a two-scale, locally conservative subgrid upscaling for elliptic problems. *SIAM Journal on Numerical Analysis*, 42(2):576–598, 2004.

[12] Todd Arbogast and Kirsten J Boyd. Subgrid upscaling and mixed multiscale finite elements. *SIAM Journal on Numerical Analysis*, 44(3):1150–1171, 2006.

[13] Todd Arbogast, Mika Juntunen, Jamie Pool, and Mary F. Wheeler. A discontinuous galerkin method for two-phase flow in a porous medium enforcing h(div) velocityand continuous capillary pressure. *Computational Geosciences*, 17(6):1055–1078, 2013.

[14] D. N. Arnold, R. S. Falk, and R. Winther. Finite element exterior calculus: from Hodge theory to numerical stability. *Bull. Amer. Math. Soc.(NS)*, 47(2):281–354, 2010.

[15] Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. Finite element exterior calculus, homological techniques, and applications. *Acta Numerica*, 15:1–155, 5 2006.

[16] Khalid Aziz, Louis Durlofsky, and Hamdi Tchelepi. Notes on petroleum reservoir simulation. 2005.

[17] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc Web page. `http://www.mcs.anl.gov/petsc`, 2016.

[18] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016.

[19] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[20] Michele Benzi, Gene H Golub, and Jorg Liesen. Numerical solution of saddle point problems. *Acta numerica*, 14(1):1–137, 2005.

[21] Alfio Borzì and Volker Schulz. Multigrid methods for pde optimization. *SIAM review*, 51(2):361–395, 2009.

[22] L. Botto. A geometric multigrid poisson solver for domains containing solid inclusions. *Computer Physics Communications, Volume 184, Issue 3, March 2013, Pages 1033-1044, ISSN 0010-4655, http://dx.doi.org/10.1016/j.cpc.2012.11.008.*

[23] A. Brandt, S. F. Mccormick, and J. W. Ruge. Sparsity and its Applications; Algebraic multigrid (AMG) for sparse matrix equations. Cambridge University Press, New York, 1984.

[24] Achi Brandt. Guide to multigrid development. In W. Hackbusch and U. Trottenberg, editors, *Multigrid Methods*, volume 960 of *Lecture Notes in Mathematics*, pages 220–312. Springer Berlin Heidelberg, 1982.

[25] Marian Brezina, Andrew J. Cleary, Robert D. Falgout, Van Enden Henson, Jim E. Jones, Thomas A. Manteuffel, Stephen F. McCormick, and John W. Ruge. Algebraic multigrid based on element interpolation (amge). *SIAM J. Scientific Computing*, 22(5):1570–1592, 2001.

[26] X. C. Cai, L. Marcinkowski, and P. S. Vassilevski. An element agglomeration nonlinear additive schwarz preconditioned newton method for unstructured finite element problems. *Applications of Mathematics*, 50:247–275, 2005.

[27] Tim Chartier, RD Falgout, VE Henson, J Jones, T Manteuffel, S McCormick, J Ruge, and P S Vassilevski. Spectral AMGe (ρ AMGe). *SIAM Journal on Scientific Computing*, 25(1):1–26, 2003.

[28] Timothy Chartier, Robert Falgout, Van Emden Henson, JE Jones, Tom Manteuffel, Steve McCormick, J Ruge, and PS Vassilevski. Spectral element agglomerate AMGe. *Domain decomposition methods in science and engineering XVI*, 55:513–521, 2007.

[29] Zhangxin Chen, Guanren Huan, Baoyan Li, Zhangxin Chen, Guanren Huan, and Baoyan Li. An Improved IMPES Method for Two-Phase Flow in Porous Media. *Transport in Porous Media*, 54(3):361–376, March 2004.

[30] Zhangxin Chen, Guanren Huan, and Yuanle Ma. *Computational Methods for Multiphase Flows in Porous Media*. Society for Industrial and Applied Mathematics, 2006.

[31] Zhiming Chen and Thomas Y. Hou. A mixed multiscale finite element method for elliptic problems with oscillating coefficients. *Math. Comput.*, 72(242):541–576, 2003.

[32] Edmond Chow, Robert D. Falgout, Jonathan J, Raymond S. Tuminaro, and Ulrike Meier Yang. A survey of parallelization techniques for multigrid solvers. Technical report, 2004.

[33] M. L. C. Christensen, U. Villa, A. Engsig-Karup, and P. S. Vassilevski. Numerical upscaling for incompressible flow in reservoir simulation: An element-based algebraic multigrid (amge) approach. *Lawrence Livermore National Laboratory Technical Report LLNL-JRNL-661295*, September 24, 2014.

[34] M. A. Christie. Upscaling for reservoir simulation. *Journal of Petroleum Technology*, 48, 1996.

[35] Xiao chuan Cai, David E. Keyes, and David P. Young. A nonlinear additive schwarz preconditioned inexact newton method for shocked duct flow. In *Proceedings of the 13th International Conference on Domain Decomposition Methods*, pages 1463–1470, 2001.

[36] K. A. Cliffe, M. B. Giles, R. Scheichl, and A. L. Teckentrup. Multilevel monte carlo methods and applications to elliptic pdes with random coefficients. *Computing and Visualization in Science*, 14(1):3–15, 2011.

[37] KA Cliffe, MB Giles, Robert Scheichl, and Aretha L Teckentrup. Multilevel monte carlo methods and applications to elliptic pdes with random coefficients. *Computing and Visualization in Science*, 14(1):3–15, 2011.

[38] M. Dumett, P. Vassilevski, and C. S. Woodward. A multigrid method for nonlinear unstructured finite element elliptic equations. *LLNL Technical Report UCRL-JC-150513*, 2002.

[39] L. J. Durlofsky, Y. Efendiev, and V. Ginting. An adaptive local-global multiscale finite volume element method for two-phase flow simulations. *Adv. Water Resour., 30:576-588*, 2007.

[40] Louis J. Durlofsky and Khalid Aziz. *Advanced Techniques for Reservoir Simulation and Modeling of Nonconventional Wells.* Aug 2004.

[41] Y. Efendiev and L.J. Durlofsky. A generalized convection-diffusion model for subgrid transport in porous media. *Multiscale Mod. Simulat., 1(3):504-526*, 2003.

[42] Stanley C. Eisenstat and Homer F. Walker. Globally convergent inexact newton methods. *SIAM Journal on Optimization*, 4(2):393–422, 1994.

[43] Robert D Falgout, Panayot S Vassilevski, and Ludmil T Zikatanov. On two-grid convergence estimates. *Numer. Linear Algebra Appl*, 12(5-6):471–494, 2005.

[44] UCD Fault Analysis Group. Saigup - sensistivity analysis of the impact of geological uncertainties on production forecasting in clastic hydrocarbon reservoirs. http://www.fault-analysis-group.ucd.ie/Projects/SAIGUP.html.

[45] Larry SK Fung, Ali H Dogru, et al. Parallel unstructured-solver methods for simulation of complex giant reservoirs. *SPE Journal*, 13(04):440–446, 2008.

[46] Michael B. Giles. Multi-level monte carlo path simulation. *Operations Research*, 56(3):607–617, 2008.

[47] Vivette Girault and Pierre-Arnaud Raviart. *Finite Element Methods for Navier-Stokes Equations: Theory and Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[48] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Matrix Computations. Johns Hopkins University Press, 2012.

[49] J.R. Wallis H. Cao, H. A. Tchelepi and H. Yardumian. Parallel scalable unstructered cpr-type linear solver for reservoir simulation. *SPE paper 96809, SPE Annual Technical Conference and Exhibition, Dallas, Texas, USA, 9-12 Oct. 2005.*

[50] W. Hackbusch. *Multigrid Methods and Applications*. Springer, 1985.

[51] Van E. Henson. Multigrid methods for nonlinear problems: an overview. *Proc. SPIE*, 5016:36–48, 2003.

[52] Thomas Y. Hou and Xiao hui Wu. A multiscale finite element method for elliptic problems in composite materials and porous media. *Journal of Computational Physics*, 134:169–189, 1997.

[53] hypre. High performance preconditioners. https://www.llnl.gov/casc/hypre/.

[54] E. Grinspun J. Bolz, I. Farmer and P. Schröder. Sparse matrix solvers on the gpu: conjugate gradients and multigrid. *ACM Trans. Graph., 22 (2003), pp. 917-924.*

[55] R. P. Kendall J. R. Wallis and T. E. Little. Constrained residual acceleration of conjugate residual methods. *SPE paper 13563, 8th Symposium on Reservoir Simulation, Dallas, Feb. 10-13, 1985.*

[56] Thiele J. Yan, F and L. Xue. A modified full multigrid algorithm for the navier–stokes equations. *Computers & Fluids vol. 36, 2007.*

[57] J. D. Jansen. The egg model - data files. *TU Delft. Dataset. http://dx.doi.org/10.4121/uuid:916c86cd-3558-4672-829a-105c62985ab2.*

[58] P. Jenny, S. H. Lee, and H. A. Tchelepi. Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. *J. Comput. Phys.*, 187(1):47–67, 2003.

[59] P Jenny, SH Lee, and HA Tchelepi. Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. *Journal of Computational Physics*, 187(1):47–67, 2003.

[60] Y. Jiang. Techniques for modeling complex reservoirs and advanced wells. *PhD Thesis, Stanford University*, 2007.

[61] J. E. Jones and P. S. Vassilevski. AMGe based on element agglomeration. *SIAM J. Sci. Comput.*, 23(1):109–133, January 2001.

[62] J. E Jones, P. S. Vassilevski, and C. S. Woodward. Nonlinear Schwarz-fas methods for unstructured finite element problems. *Second M.I.T. Conference on Computational Fluid and Solid Mechanics, Cambridge, MA, June 17-20, 2003.*

[63] Jim E. Jones and Panayot S. Vassilevski. AMGe based on element agglomeration. *SIAM J. Sci. Comput.*, 23(1):109–133, January 2001.

[64] D. Kalchev, C. Ketelsen, and P. Vassilevski. Two-level adaptive algebraic multigrid for a sequence of problems with slowly varying random coefficients. *SIAM Journal on Scientific Computing*, 35(6):B1215–B1234, 2013.

[65] D. Z. Kalchev, LEE C.-S., U. Villa, and Vassilevski P. S. Upscaling of mixed finite element discretization problems by the spectral amge method. *Lawrence Livermore National Laboratory Technical Report LLNL-JRNL-676518*, 2015.

[66] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[67] C Ketelsen, R Scheichl, and AL Teckentrup. A hierarchical multilevel markov chain monte carlo algorithm with applications to uncertainty quantification in subsurface flow. *arXiv preprint arXiv:1303.7343*, 2013.

[68] S. Knapek. Matrix-dependent multigrid homogenization for diffusion probelms. *SIAM Journal on Scientific Computing*, 20(2):515–533, 1999.

[69] S Krogstad, KA Lie, and B Skaflestad. Mixed multiscale methods for compressible flow. 2012. ECMOR XIII, Biarritz, France 10-13 September.

[70] Mohamed Sadok Lamine, Stein Krogstad, Knut-Andreas Lie, Mayur Pal, et al. Multiscale method for simulating two-and three-phase flow in porous media. *SPE Reservoir Simulation Symposium*, 2013.

[71] I Lashuk and P S Vassilevski. On some versions of the element agglomeration AMGe method. *Numerical Linear Algebra with Applications*, 15(7):595–620, 2008.

[72] Ilya Lashuk and Panayot S Vassilevski. The construction of the coarse de rham complexes with improved approximation properties. *Comput. Meth. in Appl. Math.*, 14(2):257–303, 2014.

[73] IV Lashuk and PS Vassilevski. Element agglomeration coarse raviart–thomas spaces with improved approximation properties. *Numerical Linear Algebra with Applications*, 19(2):414–426, 2012.

[74] K. Lipnikov, J. D. Moulton, and D. Svyatskiy. A multilevel multiscale mimetic (m 3) method for two-phase flows in porous media. *Journal of Computational Physics*, 227(14):6727–6753, 2008.

[75] K. Lipnikov, J. D. Moulton, and D. Svyatskiy. A multiscale multilevel mimetic (m^3) method for well-driven flows in porous media. *Procedia Computer Science, 1(1), 771 - 779, doi:10.1016/j.procs.2010.04.083*, 2010.

[76] K. Lipnikov, J. D. Moulton, and D. Svyatskiy. Adaptive strategies in the multilevel multiscale mimetic (m^3) method for two-phase flows in porous media. *Multiscale Model. Sim., 9(3), 991-1016, doi:10.1137/100787544*, 2011.

[77] P. Vassilevski M. Christensen, U. Villa. Multilevel Techniques Lead to Accurate Numerical Upscaling and Scalable Robust Solvers for Reservoir Simulation. In *SPE Reservoir Simulation Symposium, 23-25 February, Houston, Texas, USA*, pages SPE–173257–MS, 2015.

[78] D. Göddeke P. Zajac M. Geveler, D. Ribbrock and S. Turek. Efficient finite element geometric multigrid solvers for unstructured grids on gpus. *Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 419, Applied Mathematics (LS3), TU Dortmund, Germany, 2011.*

[79] H. Köstler M. Stürmer and U. Rüde. How to optimize geometric multigrid methods on gpus. *15th Copper Mountain Conference on Multigrid Methods, March 2011.*

[80] S. P. MacLachlan and J. D. Moulton. Multilevel upscaling through variational coarsening. *Water Resour. Res., 42, W02418, doi:10.1029/2005WR003940*, 2006.

[81] Kent-Andre Mardal and Ragnar Winther. Preconditioning discretizations of systems of partial differential equations. *Numerical Linear Algebra with Applications*, 18(1):1–40, 2011.

[82] D. J. Mavriplis and D. J. Mavriplis. Multigrid techniques for unstructured meshes. Technical report, in VKI Lecture Series VKI-LS, 1995.

[83] Dimitri J. Mavriplis. Multigrid approaches to non-linear diffusion problems on unstructured meshes. *Numerical Linear Algebra with Applications*, 8(8):499–512, 2001.

[84] Dimitri J. Mavriplis. An assessment of linear versus nonlinear multigrid methods for unstructured mesh solvers. *Journal of Computational Physics*, 175(1):302 – 325, 2002.

[85] Jiří Mikyška and Abbas Firoozabadi. Implementation of higher-order methods for robust and efficient compositional simulation. *Journal of Computational Physics*, 229(8):2898–2913, April 2010.

[86] J. Molenaar. Multigrid methods for fully implicit oil reservoir simulation. *TU Delft, Report 95-40, 1995.*

[87] P. Monk. *Finite Element Methods for Maxwell's Equations*. Numerical Mathematics and Scientific Computation. Clarendon Press, 2003.

[88] Joachim Moortgat, Shuyu Sun, and Abbas Firoozabadi. Compositional modeling of three-phase flow with gravity using higher-order finite element methods. *Water Resources Research*, 47(5), 2011.

[89] Olav Møyner and Knut-Andreas Lie. A multiscale two-point flux-approximation method. *J. Comput. Phys.*, 275:273–293, October 2014.

[90] M. Murphy, G. Golub, and A. Wathen. A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing*, 21(6):1969–1972, 2000.

[91] O. MÃ¸yner and K. Lie. A multiscale two-point flux-approximation method. *J. Comput. Phys.*, 275:273 – 293, 2014.

[92] S. Dalton N. Bell and L. Olson. Exposing fine-grained parallelism in algebraic multigrid methods. *SIAM J. Sci. Comput., 34(4), C123-C152, 2012.*

[93] G. Lewin D. Luebke N. Goodnight, C. Woolley and G. Humphreys. A multigrid solver for boundary value problems using programmable graphics hardware. *in HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, Aire-la-Ville, Switzerland, 2003, Eurographics Association, pp. 102-111.*

[94] NETGEN. Netgen - automatic mesh generator. http://www.hpfem.jku.at/netgen/.

[95] Society of Petroleum Engineers. Tenth spe comparative solution project. http://www.spe.org/web/csp/.

[96] M. G. Knepley P. R. Brune and L. R. Scott. Unstructured geometric multigrid in two and three dimensions on complex and graded meshes. *SIAM J. Sci. Comput., Vol. 35, No. 1, pp. A173–A191, 2013.*

[97] M. Pal, S. Lamine, K. Lie, and S. Krogstad. Validation of the multiscale mixed finite-element method. *Int. J. Numer. Meth. Fluids*, 2014.

[98] Mayur Pal, Sadok Lamine, Knut-Andreas Lie, and Stein Krogstad. Multiscale method for two and three-phase flow simulation in subsurface petroleum reservoirs. In *ECMOR XIII-13th European Conference on the Mathematics of Oil Recovery*, 2012.

[99] Joseph E. Pasciak and Panayot S. Vassilevski. Exact de rham sequences of spaces defined on macro-elements in two and three spatial dimensions. *SIAM J. Scientific Computing*, 30(5):2427–2446, 2008.

[100] S.V. Patankar and D.B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer, vol. 15, 1972.*

[101] D.W. Peaceman. Interpretation of well-block pressures in numerical reservoir simulation with nonsquare grid blocks and anisotropic permeability. *SPEJ, Soc. Pet. Eng. J.; (United States)*, Jun 1983.

[102] J. W. Ruge and K. Stuben. Algebraic multigrid (amg). *In S. F. McCormick, editor, Multigrid Methods, volume 3 of Frontiers in Applied Mathematics, pages 73-130. SIAM, Philadelphia, PA,*, 1987.

[103] B. Khailany M. Garland S. W. Keckler, W. J. Dally and D. Glasco. Gpus and the future of parallel computing. *IEEE Computer Society, September/October 2011.*

[104] Y. Saad. *Iterative methods for sparse linear systems, Second edition, 2003.*

[105] Schlumberger. *Petrel User Manual*. 2014.

[106] Joachim Schöberl. Netgen an advancing front 2d/3d-mesh generator based on abstract rules. *Computing and Visualization in Science*, 1(1):41–52.

[107] B. K. Bergen J. E. Dendy T. Austin, M. Berndt and J. D. Moulton. *Parallel, Scalable, and Robust Multigrid on Structured Grids, T-7, MS B284, Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545.*

[108] J. A. Trangenstein and J. B. Bell. Mathematical structure of the black-oil model for petroleum reservoir simulation. *SIAM vol. 49, 1989.*

[109] Ulrich Trottenberg, Cornelius W. Oosterlee, and Anton Schuller. *Multigrid*. Academic Press, 2000.

[110] Kolev T. V. and Vassilevski P. S. Parallel auxiliary space amg solver for h(div) problems. *SIAM Journal on Scientific Computing*, 34(6):A3079–A3098, 2012.

[111] P. Vassilevski and U. Villa. A mixed formulation for the brinkman problem. *SIAM Journal on Numerical Analysis*, 52(1):258–281, 2014.

[112] P S Vassilevski. Sparse matrix element topology with application to AMG (e) and preconditioning. *Numerical linear algebra with applications*, 9(6-7):429–444, 2002.

[113] Panayot S. Vassilevski. Sparse matrix element topology with application to amg(e) and preconditioning. *Numerical Linear Algebra with Applications*, 9(6-7):429–444, 2002.

[114] Panayot S. Vassilevski. *Multilevel Block-Factorization Preconditioners: Matrix-based Analysis and Algorithms for Solving Finite Element Equations*. Springer, New York, 1st edition, 2008.

[115] Panayot S. Vassilevski. Coarse spaces by algebraic multigrid: Multigrid convergence and upscaling error estimates. *Advances in Adaptive Data Analysis*, 3(1-2):229–249, 2011.

[116] Panayot S. Vassilevski and Umberto Villa. A block-diagonal algebraic multigrid preconditioner for the brinkman problem. *SIAM J. Scientific Computing*, 35(5), 2013.

[117] J.R. Wallis. *Incomplete Gaussian elimination as a preconditioning for generalized conjugate gradient acceleration*, volume SPE 11265. Nov 1983.

[118] T.C. Wallstrom, S. Hou, M.A. Christie, L.J. Durlofsky, D.H. Sharp, and Q. Zou. Application of effective flux boundary conditions to two-phase upscaling in porous media. *Transport in Porous Media*, 46(2-3):155–178, 2002.

[119] R. M. Younis, H. A. Tchelepi, and K. Aziz. Adaptively-localized-continuation-newton: Reservoir simulation nonlinear solvers that converge all the time. *SPE 119147, Stanford University, SPE Reservoir Simulation Symposium, Woodlands, Texas, USA, 2-4 February 2009.*

[120] H. Zhou, S. H. Lee, and H. A. Tchelepi. Multiscale finite-volume formulation for saturation equations. *Society of Petroleum Engineers. doi:10.2118/119183-PA*, 2012.

[121] H. Zhou and H. A. Tchelepi. Two-stage algebraic multiscale linear solver for highly heterogeneous reservoir models. *SPE Journal*, 17(02):523–539, 2012.